Foveated Stereo Video Compression for Visual Telepresence

by

Stanley Fok

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2002

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis focuses on the design of a foveated stereo video compression algorithm for visual telepresence applications. In a typical telepresence application, a user at the local site views real-time stereo video recorded and transmitted from a robotic camera platform located at a remote site. The robotic camera platform tracks the user's head motion producing the sensation of being present at the remote site.

The design of the stereo video compression algorithm revolved around a fast spatio-temporal block-based motion estimation algorithm, with a foveated SPIHT algorithm used to compress and foveate the independent frames and error residues. Also, the redundancy between the left and right video streams was exploited by disparity compensation. Finally, position feedback from the robotic camera platform was used to perform global motion compensation, increasing the compression performance without raising computation requirements.

The algorithm was analysed by introducing the above mentioned components separately. It was found that each component increased the compression rate significantly, producing compressed video with similar compression and quality as MPEG2. The implementation of the algorithm did not meet the real-time requirements on the experiment computers. However, the algorithm does not contain any intrinsic delays. Therefore, given faster processors or optimized software implementation, the design should be able to run in real-time.

Acknowledgements

I would like to thank my two supervisors David Wang and George Freeman for giving me guidance, direction and inspiration throughout my entire time here. Without their ideas and discussions, I would still be spinning my wheels somewhere in Chapter 3.

Credit also has to be given to my entire research group for being there with me during every weekly group meeting no matter how long it lasted.

My mom and dad have always been supportive of me. Thanks for all the help, especially the great home cooking and power tools. For all my family and friends who each have influenced me in many ways, thank you for what you have made me become.

Finally, I would also like to express my gratitude for the Ontario Graduate Scholarship Program and the University of Waterloo for providing me with the very important financial assistance during my graduate studies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Manipulating hazardous materials or reaching inaccessible locations using remotely operated robots is common in the today's world. Many robots have cameras mounted on them to provide visual feedback to the robot operator. Single camera systems, which present a monoscopic view of the remote location, are adequate for many tasks. However, applications that require more delicate handling of materials or better estimation of distances may require greater depth perception. In this case, a pair of cameras mounted side by side at the remote location provides a stereoscopic view for the operator.

Doubling the number of cameras on the robot means doubling the already high amount of raw video that must be transmitted to the operator's site. If the robot has a direct, wired connection to the operator, there is usually sufficient bandwidth so that the raw video can be simply transmitted. However, for wireless or Internet applications, transmitting the entire raw stereo video would be impossible with today's networking bandwidth. Hence, it is necessary to compress the video before transmitting it to the operator's site.

Stereo video compression for visual telepresence is the focus of this thesis. The work presented here uses algorithms that attempt to remove redundancy in the spatial, temporal, and stereo domains of video. For visual telepresence systems, the end user is a human

operator and thus further techniques, such as foveation, can be applied to make the compressed video feel perceptually similar to the raw video. Most telepresence systems allow the operator to pan and tilt the cameras. This generally means that the orientation of the camera platform is monitored and known. This information can also be exploited for video compression.

1.1 Thesis Outline

This thesis is organized into four main chapters. Chapter 2 discusses background information on how humans perceive images, motion and depth from stereo video devices. It also examines current image and video compression algorithms and how they function and behave.

Chapter 3 describes the design of the stereo video compression algorithm. The overall encoder and decoder diagrams are first shown, followed by a detailed discussion of each component and its function in the system.

Chapter 4 presents results generated from an implementation of the stereo video compression algorithm. The effects of each component are systematically presented and analysed. Comparisons with other common standards are also performed.

Finally, Chapter 5 summarizes the experimental results, presents concluding remarks and discusses suggestions for future research.

Chapter 2

Background

Before presenting the main body of research, this chapter reviews past research in the area of image, video and stereo video compression. Properties of the Human Visual System are examined and used to define representations for images and create stereo vision effects for visual telepresence systems. Image and video compression algorithms are surveyed and special focus is paid to wavelet-based algorithms. The DWT is derived and applied to real images, and the concept of foveated imaging is introduced as a useful feature in a visual telepresence system.

2.1 Human Visual System Stimulation

Vision or sight is the perceptual experience of seeing the surrounding environment. In a visual telepresence system, an operator at the local site visually senses the environment at the remote site. To accomplish this, the telepresence device at the local site stimulates properties of the operator's Human Visual System (HVS), generating the perceptions and visual sensations of the remote site. This section describes some major properties of human vision and explains how these properties can be exploited by systems to provide our eyes

with stimuli that create sight. This thesis assumes that the system user is a human being without any physical or psychological disabilities.

The HVS consists of the eyes and the higher order brain functions that convert light information received by the eyes into a perceived image. How the HVS functions is still under considerable research, but it is well established that a sequence of slowly varying static images creates the sensation of continuous motion in a scene [1]. This property is employed in all types of video output devices, such as film projectors, television sets and computer displays. These devices output a sequence of images, or video frames to the viewer.

To represent a scene digitally, the recording device (typically a digital video camera) samples the scene at a rate. This frame rate affects the quality of the video and is typically measured in frames per second (fps). A low frame rate produces motion that is perceived as jumpy or discontinuous, whereas a high frame rate produces smooth video that seems continuous to the human observer. This frame rate is usually constant for most applications like television broadcasting. However, Internet streaming video systems often vary the frame rate depending on the speed and congestion of the network traffic [2–4].

At each sampling instant, a rectangular video frame of fixed horizontal and vertical size is generated, and represented by a matrix of light intensity elements. Each frame element is called a pixel. Each pixel is composed of either one or three light intensity elements, depending on whether monochrome or colour video is desired. This is shown in Figure 2.1. Colour video requires three components to describe each pixel because the human retina has three types of colour photo-receptor cells, called cones [5]. Each type of cone reacts differently to short, medium, and long wavelengths of light. Hence, a video display device only needs to output three primary wavelengths of light at varying intensities to create the sensation of different colours at each pixel. These three wavelengths roughly match the three primary colours (red, green, blue) used in most video systems. This is the Red-Green-Blue (RGB) colour model. Another popular colour model is the YIQ model. It separates

the luminance, or brightness, components from the chromatic, or colour, components of an image. It is related to the RGB colour model by an invertible linear transform [6]. Monochrome video only requires one value per pixel, because the same intensity value is output for each of the three primary wavelengths, generating a gray pixel. The number of bits used to represent each intensity value determines the number of gray-scales or colours available for output. Typically each pixel intensity value is represented by at least 8 bits. For example, an 8 bit per pixel (bpp) monochrome image is able to display $2^8 = 256$ gray intensities.

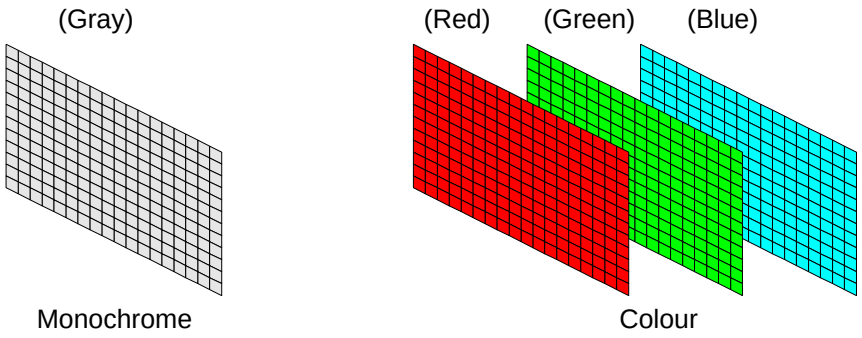


Figure 2.1: Monochrome and colour frame representation

There are other HVS properties that arise because humans have two eyes and process a stereo image representation of the scene. The fusion mechanism describes the combination of the stereo retinal images into a single percept of the scene [7]. When an observer views a scene, the retinal images in the left and right eyes are not exactly the same. Some parts of the left image and some parts of the right image match exactly. However, there are also other sections of the left image that do not exactly match, or are not viewable in the right image, and vice versa. An area that is present in the one image but not the other is called an *occluded* area. Despite the differences between the left and right retinal images, people experience a single view of the scene.

Direction and disparity are two object characteristics that are usually different in the

left and right retinal images, yet the perceived characteristics have only one interpretation. Ogle [8] states that direction is fused because direction is perceived with respect to the body, not with respect to the individual eye. Disparity, defined as the distance between corresponding points of an object in the left and right images, is also fused by a sequence of complex neural functions [7].

What is gained from stereo vision are the extra direction and disparity cues that aide the HVS in estimating the depth of objects. Note that this is not an exclusive relationship between depth and stereo vision, since the HVS simultaneously use other methods to determine depth. For example, the sense of depth is also greatly influenced by the process of motion parallax [7]. This process observes that given two objects at different distances, the farther object seems to move (with respect to the closer object) with the motion of the head, whereas the closer object seems to move (with respect to the farther object) opposite to the direction of head motion. In contrast, with a stereo image pair, the relative depths of objects are perceived because of the differences between the left and right images. According to Diner and Fender [7], "In order to achieve the percept of stereoscopic depth, the visual cortex must match up the corresponding parts of the two images and measure the binocular disparities of the matched parts of the images." Using the disparity information, the HVS can estimate the relative depth of objects.

Recording stereo video is accomplished using a dual camera system. Two cameras pointing in the same direction, are aligned along the same horizontal plane and positioned at some distance apart from each other. This distance is usually the about the same as the distance between the left and right eyes of a human being. Historically, there have been two main ways to display and store stereo video: a single combined left and right image stream or two separated left and right image streams. In combined left and right image streams, a large display device outputs the left and right video frames in an alternating manner, while the user wears special glasses that are synchronized to only allow the left eye to see

the left frames and the right eye to see the right frames. Bos [9] experimented with two types of glasses: active and passive. Active glasses physically cover, using electro-optical shutters, the appropriate eye in synchronization with the currently displayed video frame. Bos also investigated passive glasses that have different polarizations in the left and right lenses. In this case, the display device must be modified to output the left and right frames with different polarized light. For separated left and right streams, a Helmet Mounted Display (HMD) can be used. Kim *et al* [10] and Turner [11] both use dual small video displays mounted on a helmet positioned in front of the eyes. These devices are typically used in virtual reality research and entertainment. Recently there has been development in autostereoscopic devices that do not require the use of any glasses or HMDs [12, 13]. The user views the stereo video on a special holographic surface. Autostereoscopic displays use the motion parallax effect to incite the sensation of depth, not the disparity cue method.

2.2 Video Compression Motivation

Stereo video can be recorded, stored, transmitted and displayed using devices that range from simple polarized glasses to complex holographic displays. As with monocular video, the vast amount of video data to be processed is usually too large for direct storage or transmission by current hardware and software technology [14]. In ideal visual telepresence applications, the stereo video would be transmitted in real-time, and with zero delay, from the remote site to the local site. However, in real world applications, there are bandwidth limitations introduced by the networks that connect the two sites. Current transmission rates in bits per second (bps) for Internet and network traffic range from 100 Mbps (100baseT ethernet) to 56 Kbps (phone line modems). An uncompressed 320×240 resolution, 30 fps, 8 bpp (monochrome) stereo video stream would require the network to support a transmission rate of (320×240 pixels/frame)×(30 fps)×(8 bpp)×(2 streams) ≈ 36.8 Mbps. Clearly, most

current network transmission rates are unable to handle massive quantities of uncompressed video data for real-time visual telepresence. The amount of video data *must* be reduced by compressing the video stream before transmission.

Digital video compression is a specific instance of what the communications field calls *source coding*. Figure 2.2 depicts a generic compression system. As with source coding, digital video compression/coding can be classified as *lossless* or *lossy*, depending on the amount of tolerable error [15]. Assuming perfect operation in the rest of the system, lossless compression stipulates that the original data must be decompressed/decoded without any error. Entropy coding algorithms such as Run-Length, Huffman, Arithmetic, and Lempel-Ziv coding, are used for lossless data compression [15]. These coding algorithms have typical maximum compression ratios of 1.5:1 to 3:1, depending on the entropy of the source. That is, they can compress data to $\frac{2}{3}$ to $\frac{1}{3}$ of the original size. To increase the compression ratio, the perfect reconstruction criterion is relaxed in lossy compression. Lossy schemes can achieve significant compression ratios by allowing the data to be recovered with an acceptable amount of error. What is deemed as acceptable depends on the application. In video compression algorithms, typical lossy compression ratios vary but are generally over 30:1, depending on the desired quality of the output video. Notice that only lossy compression schemes can compress streamed video to the required bandwidth of current networks. The remainder of this section describes lossy algorithms only.

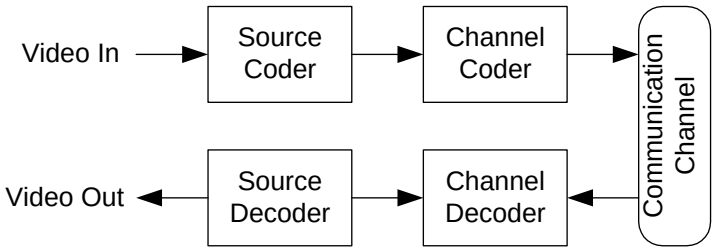


Figure 2.2: Generic compression system

The tradeoffs for increasing compression rates are an increased encoder and decoder

complexity, and an increased coding delay [15]. Coding delay is the time it takes to encode and decode a video source. A complex compression scheme usually implies a long coding delay. This delay, in addition to the network transmission delay, becomes problematic when the video should appear in real-time. For visual telepresence systems, a video delay between the remote site and the local sight diminishes the feeling of visual "presence." So far there have been no proposals that can compensate for this video delay. The only viable option at this time is to minimize the video delay by using fast software algorithms, more powerful processors or customized hardware implementations. Some work has been done at the University of Waterloo on compensating delayed teleoperation systems through the use of Kalman filtering [11, 16, 17], but this work cannot directly compensate for the video delay. For lossy compression, there is also an increased degradation in video quality as the compression rate rises. The goal is to compress the video as much as possible, while still providing an acceptable level of quality. These lossy compression tradeoffs are shown in Figure 2.3.

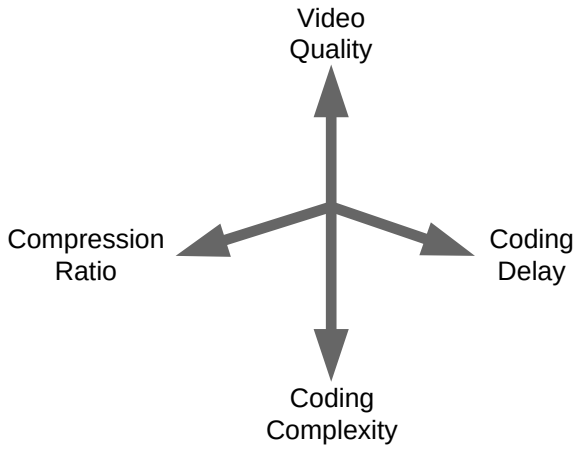


Figure 2.3: Lossy compression tradeoffs

2.3 Image Compression and the Wavelet Transform

Since video is composed of a sequence of still images, it is logical to study image compression first. It also makes sense to look at image compression algorithms because they are often used within many video compression schemes where it is called *intraframe* coding. This section presents a literature survey of popular image compression algorithms, specifically discussing Wavelet compression techniques in more detail.

2.3.1 Early Image Compression

One of the simplest ways to compress an image is to *subsample* the image data. Only a subset of the original data is stored or transmitted to the decoder. To recover the image, the remaining pixels can be upsampled by duplication. This method is simple and fast, but usually generates subjectively poor and blocky images [11]. However, in many colour image applications, this data reduction technique is commonly used. Experiments have shown that the human eye has a higher sensitivity to luminance than to chromatic perception [6]. Therefore, the chromatic components of video can be subsampled without being detected by human viewers. One of the reasons why the YIQ colour model is popular is because the chromatic components are directly available for subsampling. All standards by the Motion Picture Experts Group (MPEG) and the International Telecommunications Union (ITU), use the YIQ colour model to represent colour pixels. This allows the option of using colour subsampling in their video coders/decoders (codecs) [15].

A common form of lossy image compression is transform coding, where the image pixels are first converted into another domain by an invertible mathematical transform. The purpose of this transform is to compact as much energy of the original image sample into as few coefficients as possible. Then the compacted representation can be quantised and entropy coded before it is transmitted or stored. Compression is achieved because most

coefficients are small and rounded to zero by quantisation, and only the few large coefficients are transmitted [18]. The coarseness of quantisation and which large coefficients to send depend on how much compression is desired and how much loss in quality is tolerable. To recover the original image sample, the inverse transform is then applied to the quantised coefficients. If the coefficients are quantised coarsely, the original data cannot be recovered perfectly [15].

The Discrete Cosine Transform (DCT) is the invertible transform used in most image and video compression standards, such as the Joint Photographic Experts Group (JPEG) standard, the ITU H.26x standards, and the various MPEG standards [19]. These algorithms divide the image into small blocks usually of size 8×8 or 16×16 pixels. The two-dimensional DCT is applied to each block and the result is quantised using a predefined quantisation matrix [15]. The quantised results can then be coded by using entropy coding algorithms mentioned above. The DCT is popular because its energy compacting efficiency and computational complexity tradeoff is better than many competing transforms such as the Discrete Hadamard Transform (DHT), the Discrete Fourier Transform (DFT), and the Discrete Sine Transform (DST) [15].

2.3.2 The Wavelet Transform

Since the late 1980's, DCT coding in research has been gradually replaced by Discrete Wavelet Transform (DWT) coding [20]. The DWT has gained enormous popularity and most recent image compression algorithms use the DWT in some aspect or another. Its popularity arises from its many desirable properties, not only to image compression, but also to signal processing in general. In classical signal processing, the analysed signal is transformed from the time-domain (or spatial-domain for images) into the frequency-domain by the use of a Fourier type transform, such as the DFT or DCT. In the frequency-domain however, all sense of the signal locality in time or space is lost. The DFT of a signal only

conveys that a specific frequency has occurred within the time/space bounds of the original signal. Similarly, the original signal does not express any information about the frequency components at a specific instant of time. This phenomenon, where the complete time signal and its frequency transform cannot be simultaneously localized, is described quantitatively by the Heisenberg uncertainty principle [21]. Although it is important in theoretical calculations, real-world applications usually do not need every frequency component at every time instant. In most applications, it is sufficient to decompose the signal into just a few time and frequency components.

The analysis of music is an example of an application where some frequency and time information might be desired simultaneously. A musical score shows a musician when to play a specific note (frequency). The problem of generating the score from a musical performance is much more difficult. Taking the FT of the entire performance gives all the notes that are played. However, *when* each note is played cannot be determined from the FT. An early attempt at solving this problem was the invention of the short time Fourier transform (STFT). In this method, the FT of short bounded sections of time are computed. Although this method supplies time and frequency information simultaneously, the precision is limited by the size of the chosen window [22]. Addition problems arise because the same window size is used for all frequencies. For example, very low frequencies (long periods) will not be detected by small windows, since the chosen window may not encompass even a single period of such a low frequency [21].

What is required is a transform that varies its window size depending on the frequency. This is the basis of the wavelet transform (WT). Analogous to classical Fourier analysis, where a signal is decomposed into components with different frequencies, the WT decomposes a signal into components with different frequency *and* time properties. These basis functions are called wavelets. Where as the FT represents a function as a superposition of sinusoids, the WT represents a function as a superposition of wavelets [23]. Decomposing

a signal using the WT generates a series of sub-signals that represent signal details at different scales. The WT development in this subsection is restricted to one-dimensional (1D) signals. As will be shown later, a two-dimensional (2D) WT can be easily generated from 1D WTs for transforming images.

Wavelets are time and frequency bounded functions that are generated from a single function $\psi(x)$, by dilations and translations [23]. $\psi(x)$ is called the *analysing* wavelet or sometimes the mother wavelet. The analysing wavelet must satisfy[1] $\psi(x) \in L^2$ and $\int_{-\infty}^{\infty} \psi(x)dx = 0$ for wavelet theory to be applicable [21, 24]. Given an analysing wavelet $\psi(x)$, a wavelet $\psi_{a,b}(x)$ is mathematically expressed in the continuous domain as

$$\psi_{a,b}(x) = |a|^{-\frac{1}{2}} \psi\left(\frac{x-b}{a}\right), \tag{2.1}$$

where a is called the *scaling* parameter, b is the *translation* parameter, and x is the independent continuous variable that is usually time or space ($a, b, x \in \mathbb{R}, a \neq 0$).

The continuous wavelet transform (CWT) of a function $f(x)$ is as

$$\mathcal{W}f(a,b) \equiv \langle f, \psi_{a,b} \rangle = |a|^{-\frac{1}{2}} \int_{-\infty}^{\infty} f(x)\overline{\psi\left(\frac{x-b}{a}\right)}\, dx, \tag{2.2}$$

where the bar above the ψ function represents the complex conjugate. Notice that the CWT is the inner product of $f(x)$ and the wavelet $\psi_{a,b}(x)$. It is also a function of two variables (the scale a and translation b), and therefore can be graphically shown as a three-dimensional surface or intensity map.

In practice, calculating the CWT is computationally expensive and not necessary for reconstruction of a finite energy signal [22]. In Fourier analysis, Shannon's sampling theorem allows a finite energy or bandlimited signal to be fully reconstructed from a discrete set of sampled values. The wavelet transform also has a similar concept by sampling the

[1] L^p denotes the normed spaces corresponding to the norm $\|x\|_p \triangleq (\int |x(t)|^p dt)^{\frac{1}{p}}, \quad 1 \leq p \leq \infty$

continuous transform domain variables a and b [21]. Analogous to sampling the FT at equally spaced points along the frequency axis, a discretized WT samples the a, b plane over a grid. To restrict a and b to discrete values, let

$$a = a_0^m, \qquad b = nb_0a_0^m, \tag{2.3}$$

where $m, n \in \mathbb{Z}, a_0 > 1$ and $b_0 > 0$ [24]. The scale and translation parameters are now fixed to specific scale and translation levels, indexed by the m and n variables respectively. This relationship between a and b, samples the CWT at small scales using small translation intervals, and large scales using large translation intervals. The sampling grid is shown in Figure 2.4. This overcomes the previously stated limitations of the STFT by essentially adapting the translation interval size to the current scale being analysed. Substituting (2.3) into the CWT equations (2.1) and (2.2), produces the discretized analysing wavelet and the discrete wavelet transform (DWT) shown in (2.4) and (2.5) respectively.

$$\psi_{m,n}(x) = a_0^{-\frac{m}{2}}\psi\left(\frac{x - nb_0a_0^m}{a_0^m}\right) = a_0^{-\frac{m}{2}}\psi\left(a_0^{-m}x - nb_0\right). \tag{2.4}$$

$$\mathcal{W}f(a_0^m, nb_0a_0^m) \equiv \langle f(x), \psi_{m,n}(x)\rangle = a_0^{-\frac{m}{2}}\int_{-\infty}^{\infty} f(x)\overline{\psi\left(a_0^{-m}x - nb_0\right)}\,dx. \tag{2.5}$$

Equation (2.5) is still very complex to implement in practice due to the integral, and the freedom to choose parameter values. To simplify matters and to make a fast DWT algorithm, a widely used approach is to simply set $a_0 = 2$, and $b_0 = 1$ [21]. Also, in a digital system, the original signal is already discretized (i.e. $x \in \mathbb{Z}$). With these two simplifications, what is called a multiresolution analysis (MRA) can be performed on the input signal [24]. The idea of MRA is that the DWT can be quickly computed recursively, starting from the finest scale to the coarsest scale to generate a N-level DWT, as will now be demonstrated. The N scale levels can be considered a set of subspaces of L^2 [21]. If the finest scale (original

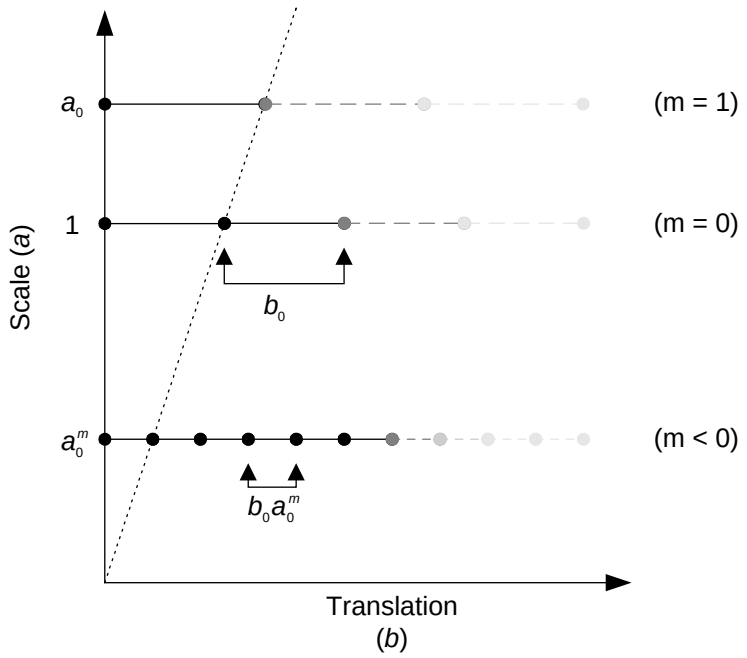


Figure 2.4: WT sampling grid

signal) is taken as the first subspace ($m = 0$), then the set of scale level subspaces can be written as $... \subset V_m ... \subset V_2 \subset V_1 \subset V_0 \subset L^2$. To recursively calculate the scale subspaces and ultimately the DWT, the auxiliary *scaling function* $\phi(x)$ is introduced along with its discretized, dilated and translated versions $\phi_{m,n}(x)$. An acceptable scaling function must satisfy[2] the recursive *scaling equation* (2.6), where h_k are coefficients uniquely defining the scaling function [21,24]. The $\phi_{m,n}(x)$'s, shown in (2.7), are basis functions of the V_m spaces. Applying (2.6) to (2.7) gives a recursive equation for the $\phi_{m,n}(x)$ basis functions shown in (2.8).

$$\phi(x) = 2^{\frac{1}{2}} \sum_k h_k \phi(2x - k). \tag{2.6}$$

$$\phi_{m,n}(x) = 2^{-\frac{m}{2}} \phi(2^{-m}x - n). \tag{2.7}$$

[2]In addition, $\phi(x)$ must also satisfy $\phi(x) \in L^2$, $\int \phi(x)dx = 1$ and $\int \phi(x)\overline{\phi(x-k)}dx = \delta(k)$.

$$\begin{aligned}\phi_{m,n}(x) &= 2^{-\frac{m}{2}}\phi(2^{-m}x - n) \\ &= 2^{-(m-1)/2}\sum_k h_k\phi(2^{-(m-1)}x - 2n - k) \\ &= \sum_k h_k\phi_{m-1,2n+k}(x).\end{aligned} \tag{2.8}$$

However, the DWT equation (2.5) requires an orthonormal basis (namely the $\psi_{m,n}(x)$ functions) of L^2. The $\phi_{m,n}(x)$'s cannot be used as an orthonormal basis of L^2 since any V_{m-1} includes V_m [21]. To obtain a recursive equation for the $\psi_{m,n}(x)$ basis functions, a new set of orthogonal subspaces ($W_m \mid m \in \mathbb{Z}, m \geq 0$) are defined. Each subspace W_m in this new set contains the "difference" in space between V_m and V_{m-1}. Specifically, $V_{m-1} = V_m \oplus W_m$, where V_m is orthogonal to W_m (i.e. W_m is the orthogonal complement of V_m in V_{m-1}) [21]. The concatenation of all the W_m subspaces is equivalent to the L^2 subspace, and therefore, finding basis functions to the W_m subspaces is equivalent to finding basis functions for L^2. Because of the relationship between the V_m and W_m subspaces, each scaling function corresponds to an analysing wavelet through (2.9), where g_k are coefficients that will be discussed later [21].

$$\psi(x) = 2^{\frac{1}{2}}\sum_k g_k\phi(x - k). \tag{2.9}$$

Applying (2.9) followed by (2.6) to (2.4) gives a recursive equation for the $\psi_{m,n}(x)$ basis functions of L^2.

$$\begin{aligned}\psi_{m,n}(x) &= a_0^{-\frac{m}{2}}\psi\left(a_0^{-m}x - nb_0\right) \\ &= 2^{-(m-1)/2}\sum_k g_k\phi(2^{-(m-1)}x - 2n - k) \\ &= \sum_k g_k\phi_{m-1,2n+k}(x).\end{aligned} \tag{2.10}$$

Before deriving the recursive equation for (2.5), let the inner product of $f(x)$ and $\phi_{m,n}(x)$

be given by the recursive equation

$$\begin{aligned} a_{m,n} \equiv \langle f(x), \phi_{m,n}(x) \rangle &= \sum_k \overline{h_k} \langle f(x), \phi_{m-1,2n+k}(x) \rangle \\ &= \sum_k \overline{h_k} a_{m-1,2n+k}, \end{aligned} \tag{2.11}$$

where[3] $a_{0,\cdot} = f(\cdot)$ [21]. Applying (2.10) and (2.11) to (2.5), gives the recursive formula for the DWT coefficients (also known as the detail coefficients).

$$\begin{aligned} d_{m,n} \equiv \langle f(x), \psi_{m,n}(x) \rangle &= \sum_k \overline{g_k} \langle f(x), \phi_{m-1,2n+k}(x) \rangle \\ &= \sum_k \overline{g_k} a_{m-1,2n+k}. \end{aligned} \tag{2.12}$$

In summary, (2.11) and (2.12) recursively calculate the DWT at each scale level starting from the original discrete signal $f(x)$. This is graphically shown in Figure 2.5. There is still the matter of the h_k and g_k coefficient values. By (2.6) and (2.9), these coefficients determine the scaling function and analysing wavelet. Furthermore, these coefficients are uniquely related to the corresponding analysing wavelet and scaling function [21, 24]. After choosing an analysing wavelet, only the corresponding h_k and g_k coefficients are required for the DWT computation - the actual analysing wavelet and scaling function are not explicitly used.

To reconstruct the original signal from its DWT coefficients, the inverse DWT (IDWT) is performed. When performing the DWT, the transform essentially replaces the basis $\phi_{m-1,n}(x)$ of V_{m-1} with the basis $\phi_{m,n}(x) \cup \psi_{m,n}(x)$ at every recursive step [21]. The IDWT reverses this transformation by taking the inner product of the coefficient weighted

[3]The $\cdot$ notation represents all values of n at scale level m, since the range of n differs for each scale level m. At the lowest scale level, the sequence $a_{0,n}$ is set equal to the sequence of $f(x)$.

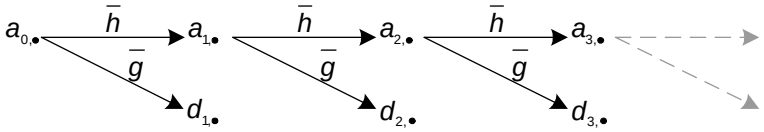


Figure 2.5: DWT recursive computation

$\phi_{m,n}(x) \cup \psi_{m,n}(x)$ basis with the $\phi_{m-1,n}(x)$ basis as shown in (2.13) and Figure 2.6.

$$\begin{aligned} a_{m-1,n} &= \left\langle \sum_k a_{m,k}\phi_{m,k}(x) + \sum_k d_{j,k}\psi_{m,k}(x) \quad , \quad \psi_{m-1,n}(x) \right\rangle \\ &= \sum_k a_{m,k}\langle \phi_{m,k}(x), \phi_{m-1,n}(x)\rangle + \sum_k d_{j,k}\langle \psi_{m,k}(x), \phi_{m-1,n}(x)\rangle \\ &= \sum_k h_{n-2k}a_{m,k} + \sum_k g_{n-2k}d_{j,k}. \end{aligned} \tag{2.13}$$

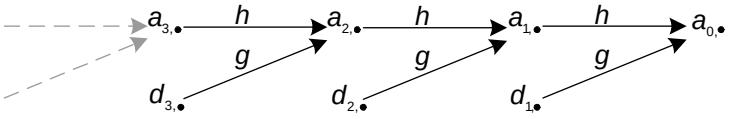


Figure 2.6: IDWT recursive computation

Since the recursive DWT and IDWT equations are essentially convolutions, digital filters can implement the transform [24]. A 1-level 1D DWT and IDWT are graphically shown in Figure 2.7. The $a_{m,n}$ and $d_{m,n}$ outputs are relabeled the L and H outputs respectively, since $\overline{h}$ and $\overline{g}$ are implemented as lowpass and highpass filters [24]. The $a_{m,n}$ coefficients are essentially the low frequency components and the $d_{m,n}$ coefficients are the high frequency components of the input signal. A higher level DWT can be computed by applying the DWT block recursively to the L output. At each sucessive level, the H wavelet coefficients

represent the next highest frequency components of the input signal. This implies that the H coefficients in the first recursive DWT stage represent the highest frequency components, while H coefficients at sucessive stages are lower frequency components. A similar reverse setup calculates a N-level IDWT. To keep the overall DWT and IDWT input/output lengths equal, the 2↓ block subsamples and the 2↑ block duplicates their inputs.

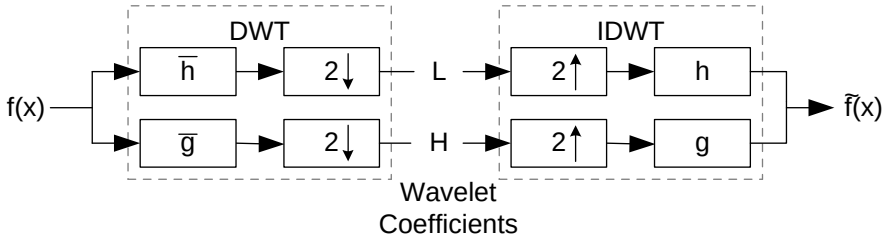


Figure 2.7: 1D DWT and IDWT

To perform a 2D wavelet transform for images, 1D transforms are simply cascaded in the form shown in Figure 2.8 [23]. Every row of the input image is first individually 1D transformed. Then the output of the first stage is 1D transformed in a column wise fashion. The 2D IDWT is performed using the same operations in reverse order. The output of the 1-level 2D DWT of Figure 2.8 is usually graphically arranged as in Figure 2.9(a). The value at each coefficient is displayed by the gray-scale intensity of the DWT image; dark pixels are low values and white pixels are high values. Similar to the 1D case, the image can be decomposed to N-levels by repeatedly applying the 2D DWT to the LL subband at each level. The recursive nature of the N-level DWT is evident by comparing Figure 2.9(a) with the 3-level DWT format in Figure 2.9(b). A typical 3-level DWT of the "Army" image in Figure 2.10(a), is shown in Figure 2.10(b). Note that the LH, HL and HH subbands in Figure 2.10(b) have been artificially amplified (multiplying by a constant factor) to show more subband detail for printing.

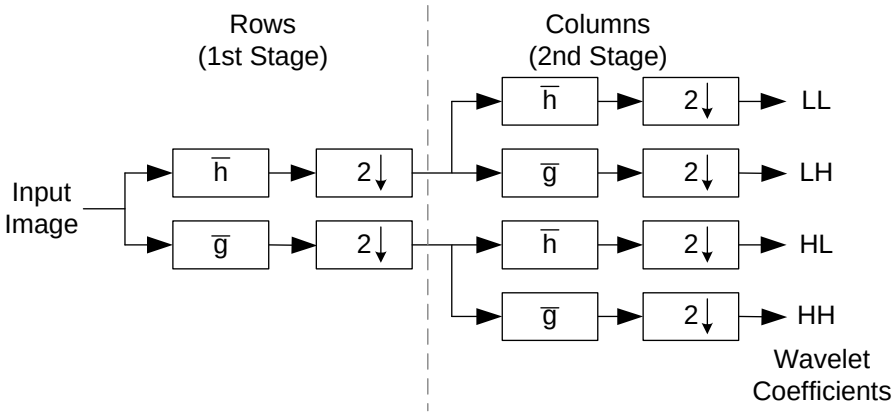
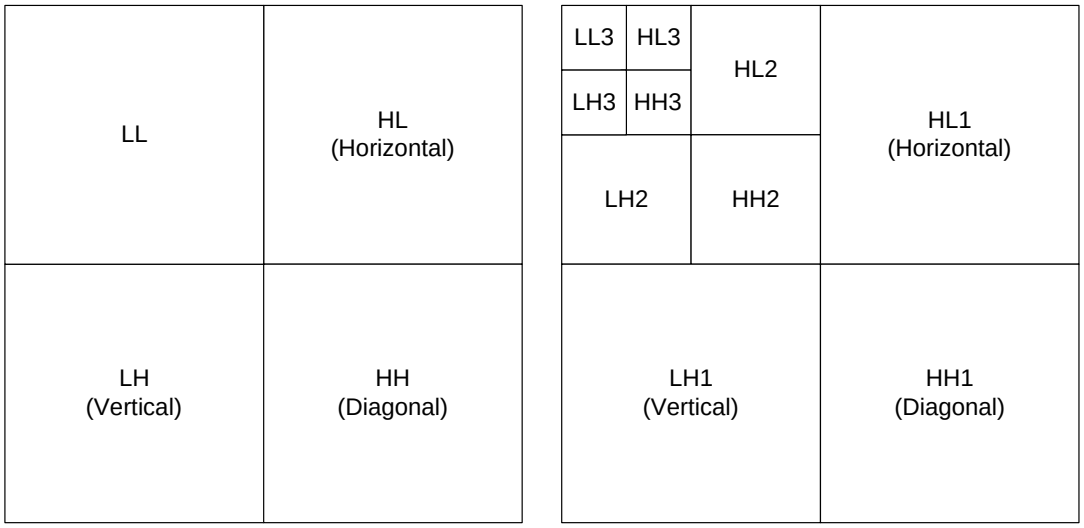


Figure 2.8: 2D DWT



(a) 1-level 2D DWT graphical arrangement (b) 3-level 2D DWT graphical arrangement

Figure 2.9: 2D wavelet decomposition arrangement

2.3.3 Wavelet-Based Image Compression Algorithms

The DWT is popular for image compression algorithms mainly because of its high energy compaction, its ability to process the image at different scales, and its reconstruction of images which are not blocky as in DCT coding. To compress an image, many proposals such as in [23, 25] apply a N-level DWT to an entire image, and then different subbands are

(a) Original "Army" real image

(b) 3-level DWT of "Army" image

Figure 2.10: Wavelet decomposition of a real image[1]

[1]Original "Army" image courtesy The Department of National Defence.

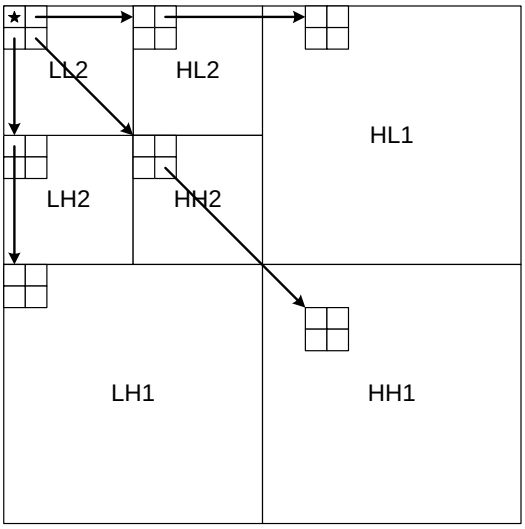
quantised differently and entropy coded. These methods work well because more bandwidth (more bits) can be devoted to perceptually important subbands and less bandwidth to the subbands that are not as important. These schemes are similar to methods used for DCT coding.

Much more impressive results can be obtained using bit plane transmission algorithms. The general concept of these algorithms is to transmit the large coefficient bits first; hence the receiver decodes significant components of the image first. These algorithms do not require any decoder training, pre-stored tables or codebooks, nor any prior knowledge of the source image [26]. At the time of their invention in the mid 1990's, these algorithms matched or outperformed all other image compression schemes that existed and continue to be the basis of many current algorithms.

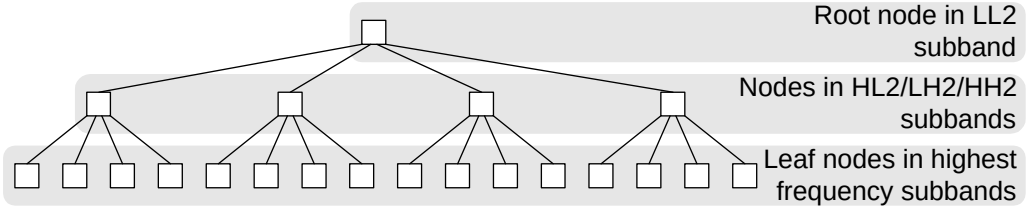
The first major contribution to this area was Shapiro's embedded zerotree wavelet (EZW) algorithm presented in [26]. The EZW algorithm used a tree structure to relate coefficients in one subband to coefficients in the next higher frequency subband. Said and Pearlman's set partitioning in hierarchical trees (SPIHT) algorithm is similar to EZW, but produces better results [27]. It is also based on a tree representation of a N-level DWT.

Consider again the 3-level wavelet transform of Figure 2.9(b). The dimensions of the subbands of every wavelet resolution level are one half of the dimensions of the previous level. This means that 1 coefficient in LH3, corresponds to 4 coefficients in LH2. The 4 coefficients in LH2 can be considered the children of the parent coefficient in LH3. Each of the 4 corresponding children in LH2, in turn relate to 4 coefficients in LH1. By repeating this pattern, one can generalize this relation to any N-level wavelet decomposition. The only exceptions to the wavelet tree structure are the coefficients in the LL subband, and the highest frequency subbands. The coefficients in the LL subband have either no children (the star coefficient in Figure 2.11(a)) or 4 children. The coefficients in the highest frequency subbands (LH1, HL1, HH1, in Figure 2.9(b)) have no children. Therefore, the coefficients

in the LL subband that have children are each roots of a different tree structure and the coefficients in the highest frequency subbands correspond to the leaves of the tree structure. The set of these trees represents the wavelet transform. Figure 2.11 shows the tree structure relationship used in the SPIHT algorithm.



(a) Parent-offspring relationship in SPIHT algorithm



(b) Tree structure for one root coefficient

Figure 2.11: Wavelet decomposition and tree structure relationship

With this tree representation of the wavelet transform, no data has yet been lost. The next step is to perform the quantisation of the tree structures. It is desirable to transmit the most significant data first (i.e. the WT coefficients with large magnitudes.) This is accomplished by bit-plane transmission, which traverses the trees and encodes only those

coefficients that are larger than a set threshold. With this threshold value, one can now exploit the significance correlation across scale levels of the wavelet transform. This property states that if a coefficient in one subband is insignificant with respect to the current threshold, it is most probable that its children and descendants are also insignificant with respect to that threshold [27]. Hence significant coefficients are transmitted first, and the rest of the coefficients are not coded at this time. Due to this property of the wavelet transform, performing the thresholding often avoids transmitting entire branches (called zerotrees by Shapiro) of the original tree structure.

After the significant data is sent, the threshold is reduced to half its original level. Then the same thresholding procedure is performed on the coefficients that were not found to be significant in the previous pass. This algorithm can be repeated until the desired application bit rate is achieved [27]. As more of the less significant data is transmitted, the decoded image will resemble the original image more, and more closely. The algorithm is usually terminated well before the bits of all the coefficients are sent. Bit rates of 1bpp to 0.25bpp give very good subjective results [27]. The SPIHT generated bitstream can then be entropy coded for further compression.

2.4 Video Compression

Video compression extends image compression to efficiently code sequences of images. This section presents a discussion about compressing standard single camera (monocular) video and stereo video.

2.4.1 Monocular Video Compression

To compress a sequence of images, an intraframe coding can be applied independently to each frame. However, except for sudden scene changes, this method is inefficient since

adjacent frames are highly correlated. Interframe coding takes the temporal dimension into account and gives significant increases in compression rates [14]. Excluding hybrids, there are three conceptually different ways to perform video compression reported in literature: block-based motion compensation, object-based motion compensation, and 3-dimensional (3D) transform coding.

The two motion compensation (MC) methods are performed in three steps [14]. The first step is to estimate the motion between the current frame and the adjacent frames. The adjacent frames can be previous frames or future frames. The second step generates the current frame prediction using the motion estimation of the first step and the adjacent frames. The last step is to encode the error (residue) between the predicted current frame and the actual current frame to fix any MC artifacts.

Block-based MC divides each frame into blocks, again typically of size 16×16 pixels to match the DCT compression algorithm [15]. Each block in the current frame is examined and its motion with respect to adjacent frames is estimated. To estimate the block motion, the current block in the current frame is compared, by calculating an error metric such as the sum of absolute differences (SAD) or the mean squared error (MSE), with candidate blocks in the adjacent frames. The candidate block with the minimum error metric is considered the best match and a motion difference vector is generated. The set of candidate blocks depend on the search algorithm used. The most basic algorithm is the full-search algorithm where every possible block in adjacent frames is considered. This method is very time consuming and generally not used in real-time video encoders [14]. Other search algorithms, such as the three-step-search [14], and Chen *et al's* fast block matching algorithm [28] are much more efficient, but they generally do not produce as good results as the full-search algorithm. MC can also be performed in the wavelet-domain where the motion vectors are calculated in different subbands. Zafar *et al* propose a multiresolution MC algorithm in [29, 30]. This method produces very good results, but has the drawback of requiring more motion vectors

than MC in the spatial-domain.

Object-based MC is similar to block-based MC except, instead of compensating block motion, entire objects in the scene are extracted and matched. This method is much more complex since it requires that perceptual objects, such as a face or an automobile, be extracted from the video scene before computing the object's motion vector [31].

The 3D transform coding techniques do not use motion estimation. Instead, the idea is to perform the wavelet transform on the temporal domain in addition to the two spatial dimensions. Then coding can be performed using a SPIHT style coder extended to the third dimension [32]. These algorithms have been reported to obtain similar compression rates as object-based MC techniques.

2.4.2 Stereo Video Compression

The compression of stereo video streams adds another dimension to video encoding. A simple method is to compress each video stream independently, but this is inefficient since the two streams are highly correlated. In stereo video, the left and right frames are almost the same except for small horizontal offsets, also known as the disparity, in different parts of the frames [33].

All stereo image and video compression algorithms in literature code one stream independently and then code the second auxiliary stream based on predictions from the independent stream. Although this method has been proven by Perkins to be suboptimal in general for a "distortion introducing encoder operating on a memoryless stereopair source," its near optimality and similarity to interframe coding makes this structure popular for practical applications [34]. The same algorithms for monocular interframe coding can be applied to estimate the disparity of the auxiliary stream. However, these disparity compensation (DC) algorithms must be modified to take advantage of the characteristics between the two video streams. One of the assumptions commonly used is that the left and right video streams lie

on the same equipolar line [7]. This requires that the camera system be perfectly calibrated so that the left and right video frames are aligned on the same horizontal plane, eliminating the need for stereo matching algorithms to search vertically displaced candidate regions. Another difference is that in stereopair disparity, every object in the left frame is displaced with respect to the image in the right frame, whereas only a few objects in monocular video move from frame to frame [34].

Although all stereo codecs have a similar structure, there are many variations in implementation. Dinstein *et al* describes the typical stereo image compression setup in [35]. After coding the independent image, the auxiliary image is coded based on the independent image using a spatial block-based DC scheme. Additionally, they also propose to take advantage of supression property of the HVS and use a low quality image for one of the streams. Dinstein *et al* found that using a low quality image for one eye has almost no loss in perceived quality or depth perception.

Algorithms that use transform coding to compress video can also perform DC in the transform-domain. Perkins proposes a scheme similar to [35] but performs DC in the transform domain coefficients instead of in the spatial-domain [34]. Gunatilake *et al* propose a similar algorithm in [33]. They introduce the idea of a worldline frame (W-frame); a frame that is predicted from previous left and right frames. They also propose that the left and right frame streams have offset intraframe and interframe sequences. This prevents the reconstructed image quality of both eyes from degrading at the same time. The offset keeps the perceived scene quality and depth perception relatively constant by pairing low quality W-frames in one eye with higher quality independently compressed frames in the other eye.

Wavelet domain algorithms have the advantage of separating the vertical, horizontal and diagonal components into different subbands. Moellenhoff and Maier exploit this by only encoding the vertical channel since this is where most of the stereo information is contained [36]. Wavelet-based algorithms can also use multiresolution DC [37], similar to

interframe multiresolution MC described in [29, 30].

As in monocular video, object-based schemes and hybrid schemes can also be implemented. Jiang and Edirisinghe propose an interesting object driven, block-based stereo coding algorithm in [38]. Instead of transmitting motion vectors and shape coding information, both the encoder and decoder execute the same algorithm to extract that information from the previous frames. Although this increases the encoder and decoder complexity, it eliminates the need to transmit vectors and shape information.

2.5 Foveation

Another HVS characteristic that can be exploited for image and video compression is foveation. Foveation is the reduction of resolution at the peripheral regions of human sight. When a person focuses on a point of interest, the objects near this foveation point are very clear, but the bordering objects of the view are extremely blurry. An example of compressing the "Army" image using normal and foveated coding is shown in Figure 2.12. Compared to unfoveated compressed images, perceived quality in foveated compressed images is improved because more bits can be devoted to the foveation point, while fewer bits are devoted to the outlying regions [39]. Foveated video is perfect for use in a telepresence system since the end user is a human being. If implemented correctly, the human viewer will not be able to distinquish between foveated and unfoveated video that have the same quality at the foveation point.

The blurring effects of foveation are caused by the arrangement of photoreceptor cells in the eye. The cell structures in the retina give rise to space-variant sampling of the scene, causing the high-frequency information in the peripheral regions to be redundant [40]. The set of cones, rods, and ganglion cells that make up the retina, have a non-uniform density across it. Cones are used for daylight vision; rods are used for lowlight scenarios and can

(a) Compressed “Army” image

(b) Foveated compressed “Army” image

Figure 2.12: Compressed “Army” image comparison

be ignored in this analysis; ganglion cells conduct the perceived information from the cones and rods out of the eye [40]. The fovea is the central point on the retina that has the highest density of cones and ganglion cells. The density dramatically falls as the distance from the fovea increases. When a human observer looks at an image, the point of focus (foveation point) is very clear and sharp while the areas further from the foveation point are perceived to have a much lower resolution.

Early attempts at foveated video simply subsampled or reduced the bit rate in regions of the image that are farther away from the foveation point producing rather blocky results as in [39, 41]. More advanced algorithms used models based on psychological experiments and wavelet approaches. Chang and Yap presented a psychological and wavelet-based foveation method for images in [42]. This paper however does not discuss how to efficiently compress images using wavelet foveation. This is accomplished by Wang and Bovik in their proposed psychological based wavelet foveation algorithm in [40]. Their algorithm artificially amplifies the foveation point coefficients in the SPIHT algorithm giving them more priority over the peripheral regions.

2.6 Summary

This section reviewed past research in the area of image, video, stereo video and foveated compression techniques. It was shown how properties of the HVS are used to present video to users and how this information can be digitally represented and stored. A brief literature survey was conducted on past image and video compression algorithms and key contributions were highlighted, including a derivation of the DWT and how it is applied to images. The concept of foveation was also introduced as a method that can be employed effectively in a visual telepresence system.

Chapter 3

Stereo Video Compression Algorithm Design

This chapter describes the design of the stereo video compression algorithm. The system's goals and requriements are first derived based on the target application, the real-time constraints and the hardware setup. The algorithm layout itself is then discussed, showing how the system's bitrate and real-time requirements are fulfilled using a combination of intraframe, interframe, stereo and foveated compression techniques.

3.1 System Goals and Requirements

3.1.1 Application

The target application for the compression algorithm is a visual telepresence system. Figure 3.1 shows the telepresence system that this experiment is designed for. Through the use of VR goggles (or some other type of stereo display), the purpose of the system is to show the scene observed at the remote site to the local site operator. Data from sensors that monitor the operator's head orientation is recorded and transmitted to the remote site, allowing the

remote camera system to mimic the operator's current gazing direction. Simultaneously, the video data from the remote cameras is transmitted back to the local site for display. In an ideal system, all the processing and transmission of data would occur without any delay. However, in a real system the communcation channel, head orientation sensors, video capture, display devices, and data processing cause many delays. These time delays cause instability in the camera-head motion control system and also cause the human user to feel negative physiological effects that reduce the sensation of being present at the remote site [11]. Fortunately, for round-trip delays of less than approximately 100 ms, instability in the camera-head motion control system can be compensated for by using a Kalman filter prediction scheme as described in [11, 16].

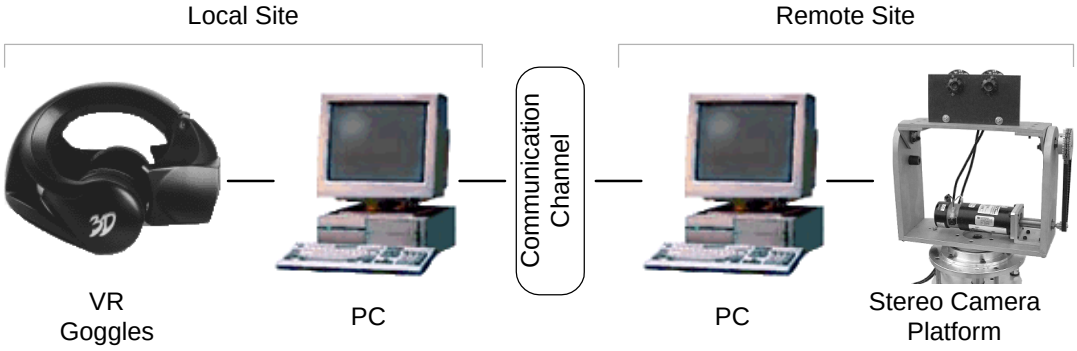


Figure 3.1: Visual telepresence system setup

Delay in video transmission is still an issue that degrades the performance of the telepresence system. It causes the user to see the scene of the remote site after it has already occurred. To date, there have been no proposals that attempt to compensate for this delay. The only alternative option is to minimize the overall video delay by designing the stereo video compression algorithm so that it has a minimal amount of coding and decoding delay. This is the strategy used for designing this stereo video compression algorithm.

3.1.2 Real-time Constraints

If the desired video frame rate is 30 fps (approximately the same as standard broadcast television in the U.S.A. and Canada [6]), the total delay per frame from the remote cameras to the local user display cannot exceed $1/30 \approx 33$ ms. Assuming that all the computations of the algorithm can be performed in real-time (either via customized hardware or optimized software), the video codec must guarantee that all inherent delays in the algorithm are removed. This means that the algorithm must be causal and cannot rely on future video frames for compression, a practice that is commonly used in the current MPEG and H.26x algorithms [15]. Only past and current video frames can be considered when generating the compressed representation of the current frame. This is the main difference between standard video compression algorithms and algorithms that are used in real-time video systems.

3.1.3 Experiment Hardware Setup

The stereo visualization device in our experiment setup is a pair of Interactive Imaging Systems VFX3D VR goggles [43]. It requires that the video displayed in each eye have a resolution of 640×240 lines in the vertical and horizontal directions respectively. Colour video is possible on the goggles, but for this experiment only monochrome video will be used to reduce the uncompressed video data size. The VR goggles have a maximum frame rate of 75 fps, but this experiment will use a frame rate of 25 fps, which is the around the maximum capture rate of the video capture cameras. Two ADS Technologies Pyro Firewire Webcams [44] are mounted 6 cm apart on a robotic pivoting platform to provide the stereo video streams required. Both cameras are calibrated so that they are equipolar, meaning that the top and bottom edges of their respective captured video frames are horizontally aligned. The camera platform is almost identical to the one used by Turner in [11], except that the camera mounting on Turner's system was upgraded to a more easily aligned and

rigid fixture. The platform has two degrees of freedom, allowing the stereo cameras to be moved in the pan (left and right) and tilt (up and down) angular directions. Encoders on the pan and tilt motors allow a computer to monitor and record the orientation of the cameras. Using this setup, each frame of captured video can be assigned a corresponding angular position.

Altogether, the algorithm is designed to compress two monochrome (8 bpp) video streams that each have a resolution of 640×240 at 25 fps. Therefore the total uncompressed bitrate in bits per second (bps) is (2 streams)$\times$(640$\times$240 pixels/frame)$\times$(8 bpp)$\times$(25 fps) = 61 440 000 bps. The processors for this system are off-the-shelf Intel Pentium III 866 MHz personal computers running the Microsoft Windows 2000 operating system. In [11], Turner designed his image compression algorithm to operate under 10 Mbps for the same video format as this experiment uses. He states that 10 Mbps is feasible for mid-range Internet connections. The goal of this algorithm is to *significantly* improve on this rate. In Turner's work, the stereo video streams were independently compressed by subsampling and using coarse quantization. Also, foveation was performed by lowering the resolution in outer image regions arbitrarily. The algorithm presented in this thesis uses more advanced compression techniques and exploits the correlation between both video streams. The design here also foveates the video using a psychologically based method that suits the HVS very closely.

3.2 Overall Design

The stereo video compression algorithm compresses one of the video streams (main stream) independently of the second stream. The second stream (auxiliary stream) is compressed based on the disparity between both streams. This design allows the video stream to be decoded by stereo and mono stream display devices. Normal mono stream displays can

simply ignore the auxiliary stream compression data and decompress the main stream only. Using this paradigm, there are several distinct components for the compression system; the intraframe, interframe and disparity compression algorithms.

The overall design of the compression algorithm is heavily influenced by the method used to compress the interframe or temporal dimension data. Given that the current frame, previous and subsequent frames are highly correlated with each other, only changes between frames need to be transmitted. As mentioned in Chapter 2, the three main ways to perform temporal compression are block-based motion compensation, object-based motion compensation and 3D transform coding. Although object-based motion compensation algorithms usually compress video more than block-based algorithms, they are not suitable for real-time applications because the algorithms are generally complex and time consuming. In the future however, these methods can be applied more widely when faster algorithms are proposed and more powerful processors made commonly available. 3D transform coding, such as 3D-SPIHT [45], must wait until a sufficient group of frames is recorded before performing the transform in the temporal dimension. This creates a time delay that cannot be eliminated and therefore makes these types of coders unacceptable for visual telepresence. The one remaining option is a block-based algorithm. These algorithms (also used in the MPEG and H.26x standards) generally can compress video sufficiently and relatively quickly without delay.

The block-based algorithm used in this experiment divides the input video sequence into sections or groups of frames (GOF). The first frame in the GOF (I-frame) is always compressed independently of any other frame. The remaining frames in the GOF (P-frames) are all compressed based on previous frames. The number of frames per GOF determines the tradeoff between video robustness and compression rate. Long GOF's allow a higher compression rate at the expense of lower error tolerance. In general I-frames cannot be compressed as much as P-frames, so fewer I-frames translates to better compression [14].

However, since all the P-frames are based on previous frames in the GOF, a single error can propagate through the entire GOF. Short GOF's have a higher tolerance for errors, but also require more bandwidth since I-frames are more numerous. Another advantage of short GOF's is better performance at sudden scene changes in the video (assuming no dynamic scene change detection is used). When the video in a scene suddenly changes, interframe coding becomes less efficient than intraframe coding [14]. After a scene change, a short GOF means less time passes before an I-frame is transmitted. However, since the application is a real-time visual telepresence system, there are no sudden scene changes and a large GOF is acceptable. During experimentation the GOF size of 25 frames was found to be subjectively acceptable.

An intraframe compression method must also be chosen to compress the I-frames and error residues from the interframe motion estimation. All MPEG and H.26x standards use block DCT coding for I-frame and residue compression [15]. This type of coding divides each frame into fixed sized blocks. Each block is transformed using the DCT and then quantised before transmission or storage. Block DCT coding seems logical since the interframe motion estimation already uses a block-based algorithm. However, this experiment also hopes to combine the benefits of foveation into the overall compression algorithm. Although fixed sized block-based DCT coding can be modified to produce foveated results, it is not the most natural choice since there is no mechanism to perform pixel accurate region-dependent image quality manipulation [41]. At low bitrates, blocking artifacts become more apparent and distracting for viewers. A more fitting and effective intraframe coder that also smoothly foveates the reconstructed images is the foveated SPIHT algorithm presented by Wang and Bovik in [40]. This method uses the wavelet transform and a SPIHT style coder to compress images. The reconstructed images do not suffer from blocking artifacts at low bitrates while at the same time performing accurate psychological foveation. The previous image in Figure 2.12(b) was obtained using the foveated SPIHT coder. All I-frames and residues will be

compressed using this method.

The compression of the auxiliary streams can be accomplished using the same block-based motion estimation algorithm chosen for the main stream's interframe compression. For added efficiency, the main stream's motion estimation algorithm needs to be modified to only consider horizontal disparities between the left and right video streams, because it is assumed that both cameras are pointing in the same direction and aligned on the same plane.

The block diagrams of the stereo video compression encoder and decoder are shown in Figures 3.2 and 3.3 respectively.

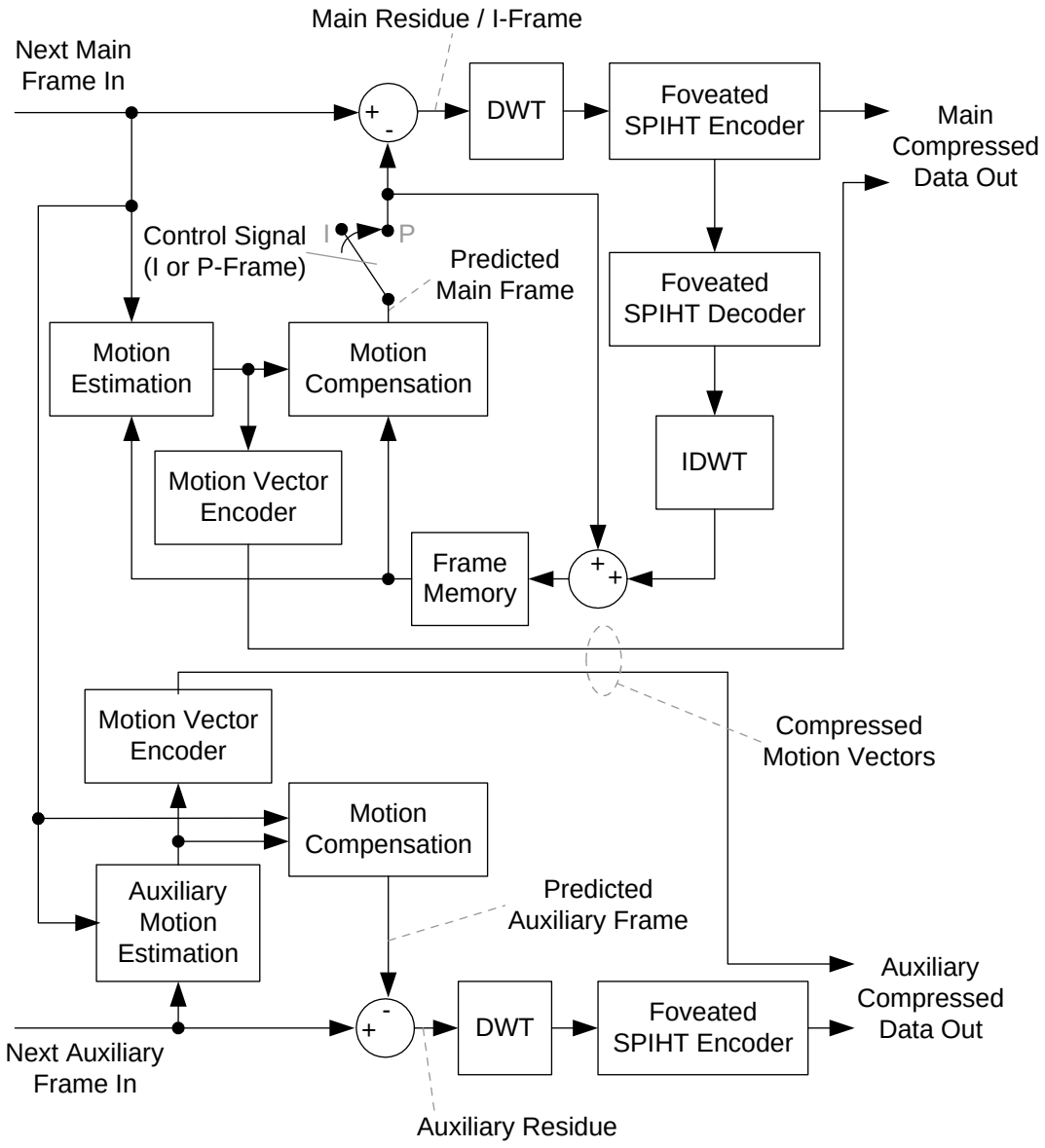


Figure 3.2: Stereo video compression algorithm encoder block diagram

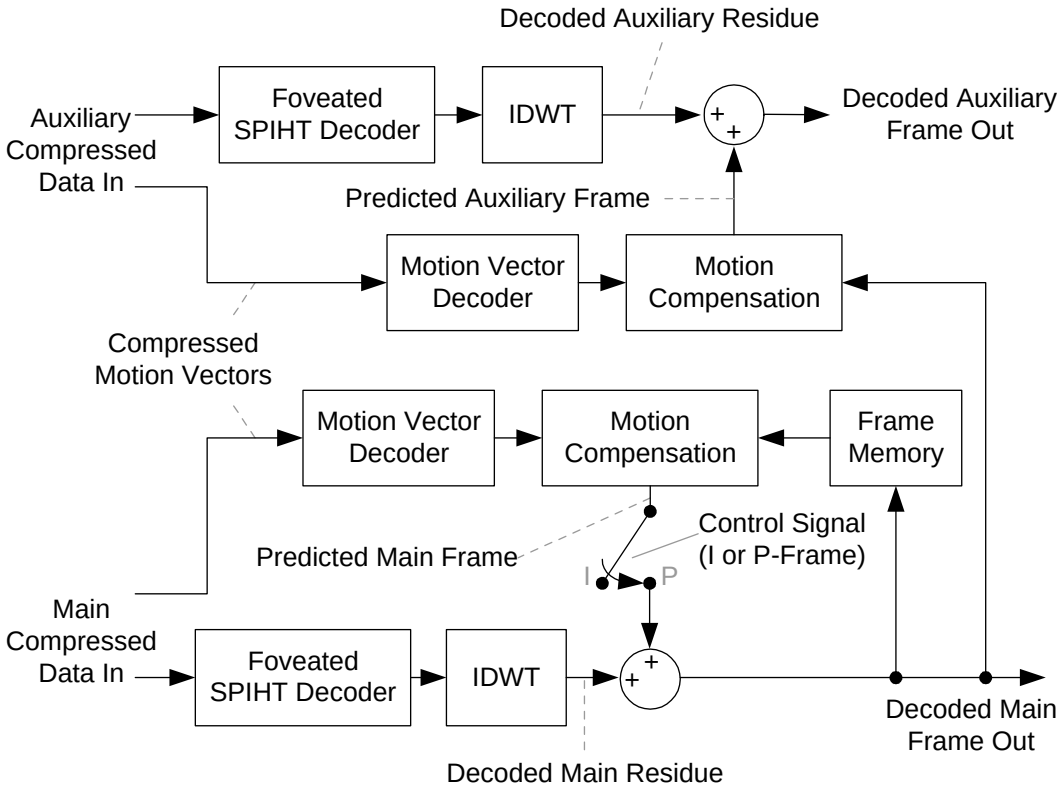


Figure 3.3: Stereo video compression algorithm decoder block diagram

3.3 Foveated Intraframe Compression

The foveated SPIHT algorithm proposed in [40] is used for intraframe compression. This wavelet based algorithm weights the DWT coefficients before applying a modified SPIHT algorithm. Upon decoding, inverse weighting and IDWT, the reconstructed image is foveated. The block diagram of the foveated SPIHT algorithm is shown in Figure 3.4. As described in Section 2.3.3 the general SPIHT algorithm takes the DWT of the input image and then sends the coefficients out in a bit plane style manner. Thus, coefficients with a larger magnitude (containing more information) are sent first. Slight changes need to be made to the SPIHT algrorithm because the dynamic range of the coefficients have been changed by the foveation weighting [40]. An example of an image compressed using the general SPIHT algorithm was shown previously in Figure 2.12(a). Notice that the loss of image details is uniform throughout the image. The foveated SPIHT algorithm instead weights the DWT coefficients before transmitting them so that coefficients corresponding to the foveation zone are amplified and transmitted earlier. The result is higher detail in the foveation area than the rest of the image, as shown in Figure 2.12(b).

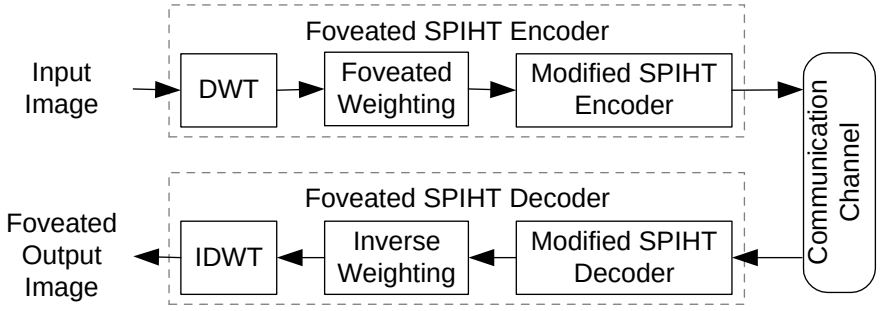


Figure 3.4: Foveated SPIHT algorithm block diagram

The weighting is based on psychological experiments that measure the eye's contrast sensitivity (sensitivity to detail) as a function of retinal eccentricity [40]. Retinal eccentricity, is the angular position in degrees from the centre of the retina (fovea). At 0 degrees the eye

has the highest concentration of cones and therefore detects more details of the scene. The concentration of cones decreases exponentially as the eccentricity increases. A mathematical model that fits the contrast sensitivity of a typical retina is described in Appendix A. The appendix also uses that model to derive a normalised wavelet-domain weighting function that generates foveation weighting masks as shown in Figure 3.5. The white regions in the figure represent magnitudes near 1 and the dark regions are magnitudes near 0. Different masks are generated depending on the parameters used in the weighting function derivation.

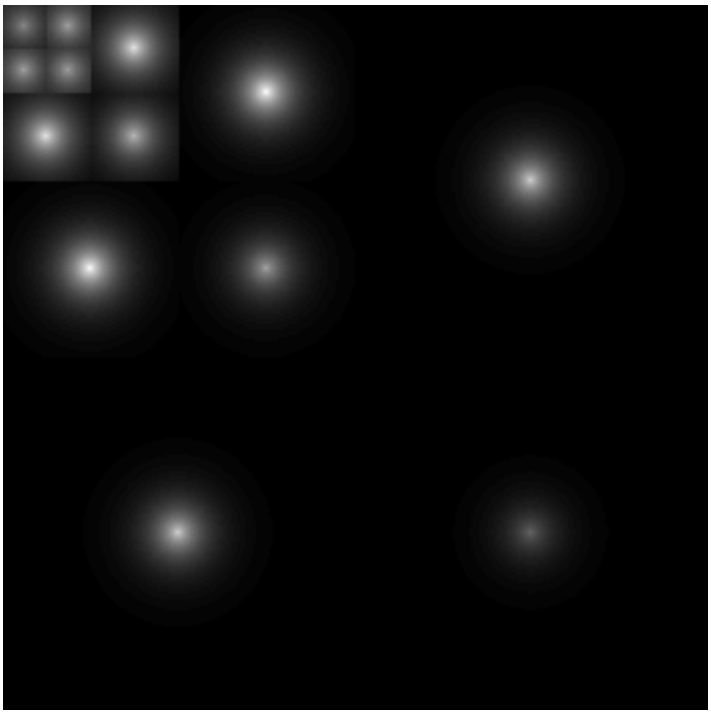


Figure 3.5: Sample foveation wavelet sensitivity function

3.4 Interframe Compression

3.4.1 Block-based Motion Estimation

The idea of interframe compression is to remove the redundancy between adjacent frames and only transmit or store the difference. For simplicity the block-based algorithm uses a fixed-sized block. As shown in Figure 3.4.1, this block-based algorithm divides the current frame to be processed into an array composed of square blocks.

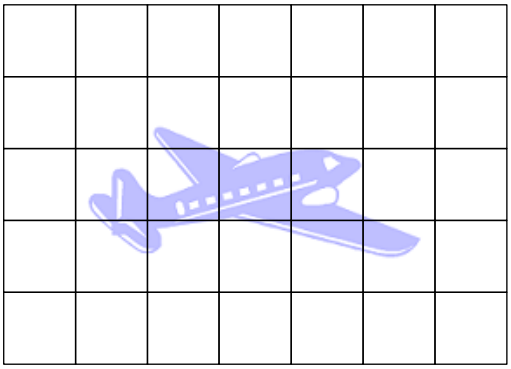


Figure 3.6: Frames divided into array of fixed-sized blocks

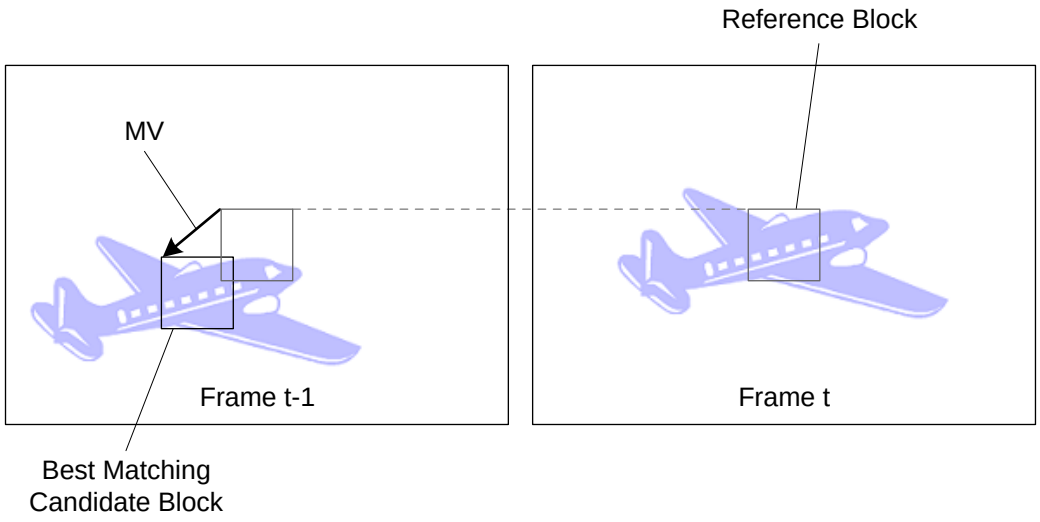
Given the current frame and for each block in the frame, the previous frame is searched to find a block-sized area (candidate block) that best matches the current block being processed (reference block). Only the immediately previous frame and not other previous frames is searched to limit computation and because with high probability, it will resemble the current frame the closest. Also future frames are not searched since they are not available in a real-time application. The matching criterion used is the sum of absolute differences (SAD) given by equation (3.1), where x is summed over all the pixels in a block. (Note Section 3.4.2 discusses how the SAD can be modified to reduce computation time, yet still give meaningful output values.) The SAD is used because it gives good matching results and takes fewer computations than other matching criterions, such as the mean square error

(MSE).

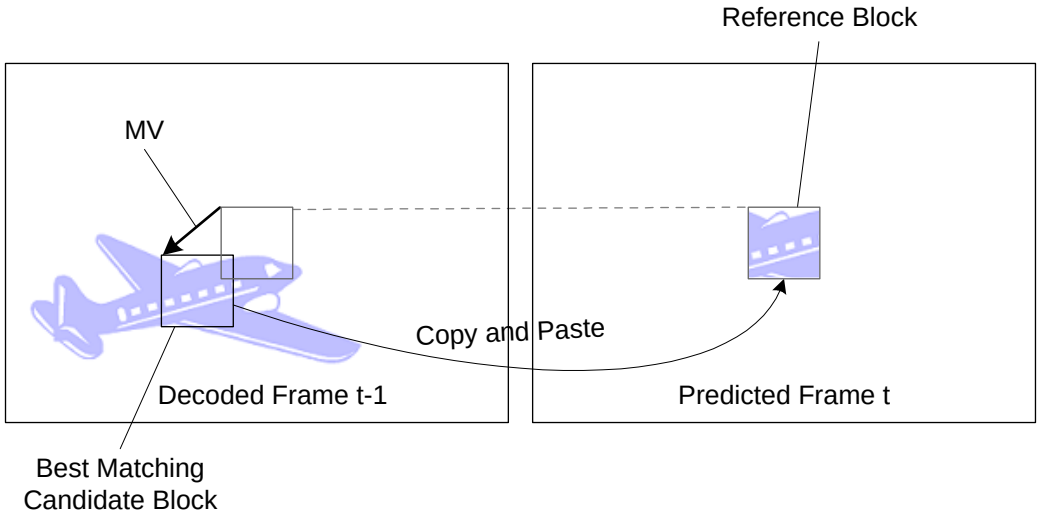
$$SAD = \sum_{x} \Big| \text{ReferenceBlk}(x) - \text{CandidateBlk}(x) \Big|. \tag{3.1}$$

The candidate block that has the smallest SAD is regarded as the best match to the reference block. The position difference (translation) between the best candidate and reference block is calculated and called the motion vector (MV) of the reference block. In this implementation the MV's are measured in full pixels for simplicity. Note that in newer MPEG and H.26x standards 1/4 and 1/8 pixel MV's are also allowed [15]. This matching process is also known as block motion estimation (ME) and is shown in Figure 3.7(a). Each block in the current frame is processed in this manner until all the reference blocks have an associated MV. The array of MV's that is created for the current frame describes the motion of the reference blocks with respect to the previous frame (i.e. the positions in the previous frame that the reference blocks have moved from). In essence, each MV replaces the data of one block. If each MV is represented by 2 bytes (one byte for each of the horizontal and vertical spatial directions) and each block is 16×16 pixels, then for 8 bpp monochrome video, the data for 1 block = 2048 bits is replaced by 2 bytes = 16 bits – a compression ratio of 128:1! As with any type of data, even more compression is achieved when the MV array is compressed using a lossless entropy coder. The MPEG and H.26x codecs use a Huffman algorithm for the entropy compression [15], but this algorithm uses the generic Lempel-Ziv algorithm called zlib, since the source code is freely distributed at [46]. It is necessary to transmit the motion vectors using a lossless algorithm because even small differences in the motion vectors will lead to incorrectly matched blocks in the decoder. Incorrectly matched blocks may introduce very large errors depending on the content of the image.

Representing each MV direction by 1 byte means that a maximum of $2^8 = 256$ distinct values can be assigned. Since MV directions can be positive or negative, the actual MV range is from -128 to +127 pixels in implementation. For comparison, most video applications find



(a) Motion estimation process



(b) Motion compensation process

Figure 3.7: Motion estimation and compensation

that a MV range of -31 to +31 pixels (6 bits per MV direction) is adequate [15]. However, this experiment uses 8 bits per MV direction since it is convenient for SW implementation. For increased compression performance, the MV's can be converted to the smaller 6-bit

representation.

At the decoder side, the current frame is reconstructed using the MV's and the previously decoded image creating a *predicted* image. As shown in Figure 3.7(b), the predicted image is created by setting each block equal to the block that the corresponding motion vector points to in the previous frame. However, the current image cannot be reconstructed solely with motion vectors since not all movement in a frame is translational. For example, Figure 3.8 shows how the rotation of a square cannot be described using the motion estimation scheme alone. To reconstruct the original image, the predicted image must be adjusted by adding the difference between the actual frame and the predicted image. This difference, called the error *residue* image, is transmitted to the decoder using the lossy foveated SPIHT algorithm. If the motion estimation algorithm performs accurately, the residue image will be mostly empty with some small areas that have error information in them. Since most of this image is approximately zero, very few bits are needed to represent this image adequately. By using the foveated SPIHT algorithm here, the foveation area of the residue image will be given priority and the decoded frame will be foveated.

Another criteria that affects image quality is the block size. In general, smaller block sizes will improve the reconstructed image quality since smaller blocks can represent the motion of objects in the video more accurately. However, a smaller block size also means each frame is divided into many more blocks in total, increasing the number of MV's required to represent the motion between frames. Larger blocks have trouble representing object motion accurately, but require fewer MV's. Fewer MV's bits also means that more bits can be allocated to the error residue. This experiment uses a 16×16 block size, which is the same as the MPEG and H.26x standards.

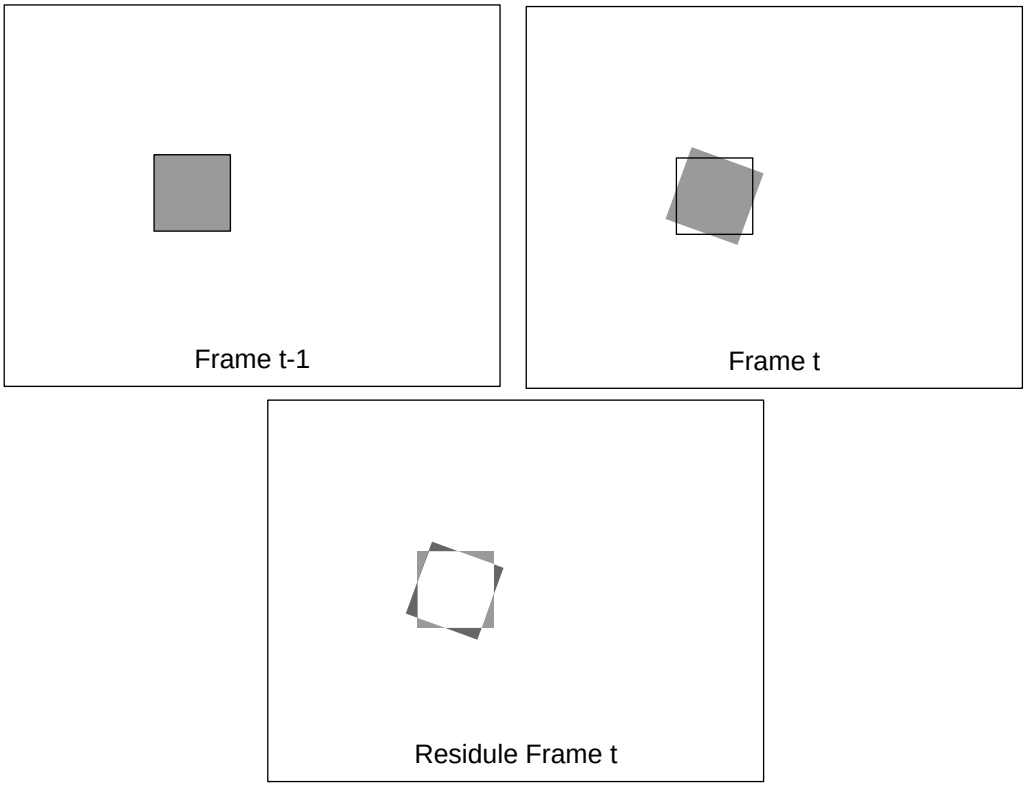


Figure 3.8: Box rotation residue

3.4.2 Fast Motion Vector Estimation Algorithm

The computationally intensive work in interframe coding occurs in the motion estimation algorithm. There are many ways to carry out motion estimation. First and most straightforward is the brute force method, where every possible position within a search window in the previous frame is tested for the best match. This windowed full-search algorithm is the most accurate, but too slow for real-time applications on a PC. To increase the speed of the matching, a fast search algorithm is performed. Some fast search algorithms are broadly categorized in the following groups [47]:

- Heuristic Search Techniques: These algorithms reduce the number of candidate MV's tested in each search area.

- Fast Matching Techniques: The matching criterion (SAD) is calculated using only a subset of the pixels in each block.

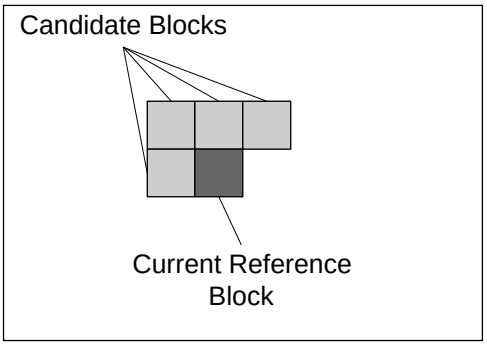
- Spatio-Temporal Correlation Techniques: Previously calculated MV's from the previous frame and current frame are used as initial candidate MV's and updated using a small refinement search.

- Hierarchical and Multiresolution Techniques: A lower resolution version of the frames are matched first and then refined until the normal resolution version of the frames is reached.

The search algorithm adopted here is a combination of the above four techniques and is based on the proposed algorithm by Chalidabhongse and Kuo [48]. The difference between that algorithm and this one is that different candidate blocks and a different refinement search are used. The basis of this algorithm is a spatio-temporal technique, which makes the assumption that neighbouring blocks of the reference block in the current and previous frame have similar motion vectors. Using this assumption, previously calculated neighbouring motion vectors are used as initial estimates when searching for the motion vector of the current reference block. A small search, or refinement pass, is then performed around the best matching MV to generate a more accurate final MV for the reference block.

The first phase in the algorithm uses previously computed MV's from the current and previous frame as initial estimates for the MV of the reference block. Ignoring the temporal dimension for now, a straightforward way to perform just spatio correlated estimation is to use as initial estimates, the MV's of the block neighbours that are *immediately* adjacent to the reference block in the current frame. If the blocks are processed in a raster scan (top to bottom, left to right), the SAD for the MV's of the blocks above, left, above-left and above right of the reference block can be calculated. This is shown in Figure 3.9(a). The MV with the minimum SAD will be more accurately processed in the refinement pass.

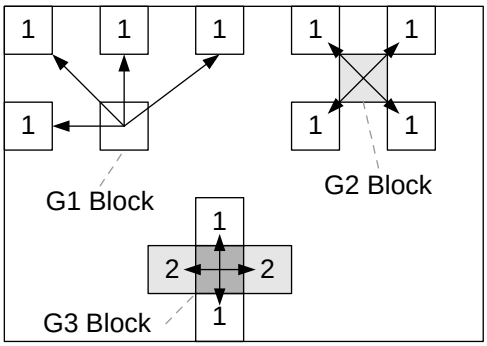
This straightforward spatio correlated estimation method inceases the speed of the search algorithm.



(a) Simple spatio candidates

1	3	1	3	1	3	1	3	1	3
3	2	3	2	3	2	3	2	3	2
1	3	1	3	1	3	1	3	1	3
3	2	3	2	3	2	3	2	3	2
1	3	1	3	1	3	1	3	1	3
3	2	3	2	3	2	3	2	3	2
1	3	1	3	1	3	1	3	1	3

(b) G1, G2, G3 block classification



(c) G1, G2, G3 spatio block candidates

Figure 3.9: Spatio correlated motion estimation candidates

However, a larger increase in speed is achievable if the algorithm takes advantage of the observation that MV's have a longer range spatial correlation than one block [48]. This means that blocks that are slightly farther away from the reference block probably also have a similar MV. For this reason the algorithm classifies every block into one of three groups; G1, G2 or G3. Figure 3.9(b) shows how each block is classified. First, all G1 blocks are processed, followed by all G2 blocks, and finally by all G3 blocks. When processing G1 blocks, only MV's of other previously processed G1 blocks can be considered as candidate MV's since the G2 and G3 blocks have not yet been processed. G2 block

processing uses adjacent G1 blocks. G3 block processing uses neighbouring G1 and G2 blocks. This candidate selection, shown in Figure 3.9(c), allows G2 and G3 blocks to have initial MV estimates coming from non-causal spatial directions, and not just from blocks that are above and left as in the simple spatio method. A faster and more accurate search is the result since a better set of initial estimates is used. This grouping also means that G3 blocks have slightly better set of initial MV's than G1 and G2 blocks, so the refinement pass for G3 blocks does not need to be as rigorous as that of the G1 and G2 blocks [48].

The above discussion focused on spatio estimation, in other words, estimating the MV using previously calculated MV's in the same frame only. MV's in the previous frame are also highly correlated. For all reference blocks, a possible candidate MV is the MV of the block at the same location in the previous frame. Also the MV's from all immediately neighbouring blocks in the previous frame can also be used. Note that temporal candidates cannot be used for the very first P-frame of a GOF since the previous I-frame has no MV array. Figure 3.10 shows the MV candidates for spatio-temporal MV estimation.

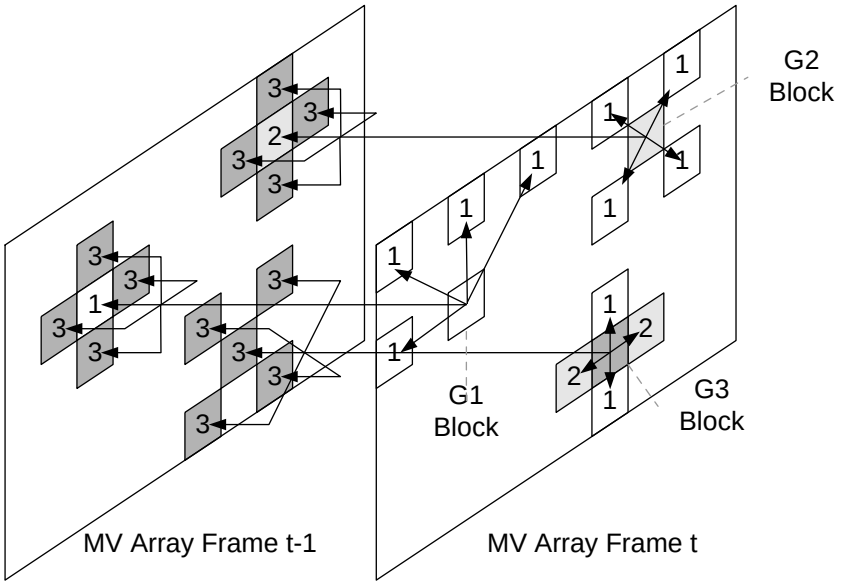


Figure 3.10: G1, G2, G3 spatio-temporal block candidates

The next phase in the spatio-temporal estimation algorithm is the refinement process. The refinement process takes the best candidate MV for each reference block and performs a small search around the area pointed to by that MV. This makes the final MV more accurate and is also the mechanism that introduces changes to values in the MV array. The search algorithm in the refinement pass is a simple three-step-search heuristic technique. The SAD is calculated for the eight MV's surrounding the current best candidate MV. If one of the surrounding MV's has a smaller SAD than the current best candidate MV, the search repeats with the new MV as the best initial MV estimate. Otherwise the search ends and the current best MV is declared the best match. For a three-step-search, this repeats three times, although any number of iterations can be used. The assumption that makes the three-step-search work is that the matching error, SAD, increases monotonically as the searching point moves away from the global minimum. This assumption is generally not true if the initial MV estimate can be any point in the frame, but is reasonable if the initial MV supplied by the spatio-temporal algorithm is already close to the global minimum [48]. Figure 3.11 shows the three-step-search procedure.

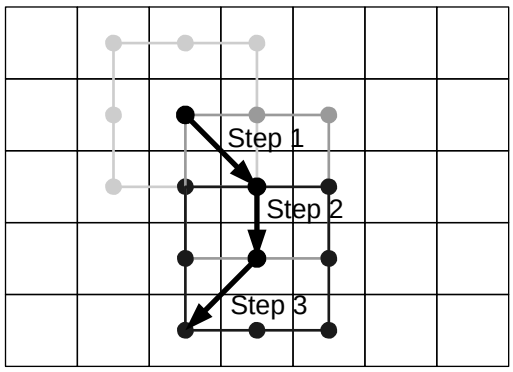


Figure 3.11: Three-step-search algorithm

To further increase the speed of computation, the spatio-temporal algorithm can be applied in an hierarchical technique. This technique creates lower resolution versions of each frame and performs block motion estimation with the lower resolution frames. Each lower resolution frame is half the size of the previous level's dimensions. MV's from the lower resolution level are used as initial estimates (after scaling by a factor of 2) for the next higher resolution level. The speed increase comes from the reduced search area and reduced block sizes at the lower resolution levels. To start the hierarchical algorithm, the original frame is decomposed into multiple low resolution images. This algorithm uses a simple and fast averaging technique, where the average of four neighbouring pixels is used as the corresponding pixel value in the next lower resolution image. Three levels of lower resolution images per frame were found to be adequate in [48] and thus is used in this experiment. At the lowest resolution (coarsest) level, a small windowed full-search is performed for each block. This full-search is fast since at this resolution level each block is only 2 pixels $\times$ 2 pixels ($16/2^3 = 2$) and the window is set to constrain the search area to +/- 1 block. At the next resolution level, the above spatio-temporal algorithm is performed using the additional candidate MV generated by the full-search of the coarsest resolution level. This is repeated for the next resolution level until the full resolution level is reached. Each resolution level uses the additional candidate MV generated by the previous level. [48] shows that using a hierarchical technique (as opposed to just the spatio-temporal technique) usually improves the speed of the algorithm and reduces the error between the decoded frames and the original frames.

The final improvement to the algorithm is a modification to the SAD equation. The previous SAD equation (3.1) sums the absolute difference over all the 256 pixels in a block at the full resolution level. However, the SAD still functions adequately if only half of the block pixels are actually used in the calculation. Figure 3.12 shows the pattern of the pixels that are actually used (shown in dark squares) in the SAD calculation. White squares are

ignored pixels. Even fewer pixels can be used, but as the number of pixels included in the calculation reduces, so does the validity of the SAD calculation.

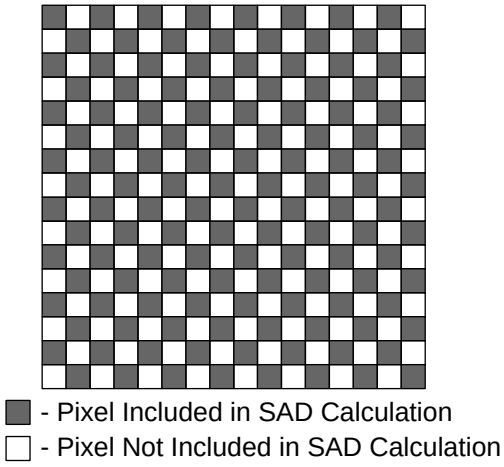


Figure 3.12: Pixel reduction in SAD calculation

3.5 Stereo Compression

For this experiment, the main stream is the right camera video and the auxiliary stream is the left camera video, although the choice of which stream is the main or auxiliary does not affect the outcome. The auxiliary stream is coded based on the main stream, and uses almost the same spatio-temporal algorithm for disparity estimation and compensation as inteframe coding uses for motion estimation and compensation. In the above Section 3.4, the main stream is compressed with the spatio-temporal algorithm which computes the SAD between blocks in the current main stream frame and the previous main stream frame. To compress the auxiliary stream, the spatio-temporal algorithm instead computes the SAD between blocks in the current auxiliary stream frame and the current main stream frame. The disparity compensated auxiliary frames will be called S-frames. Figure 3.13 shows the overall relationships between the frames in both video streams.

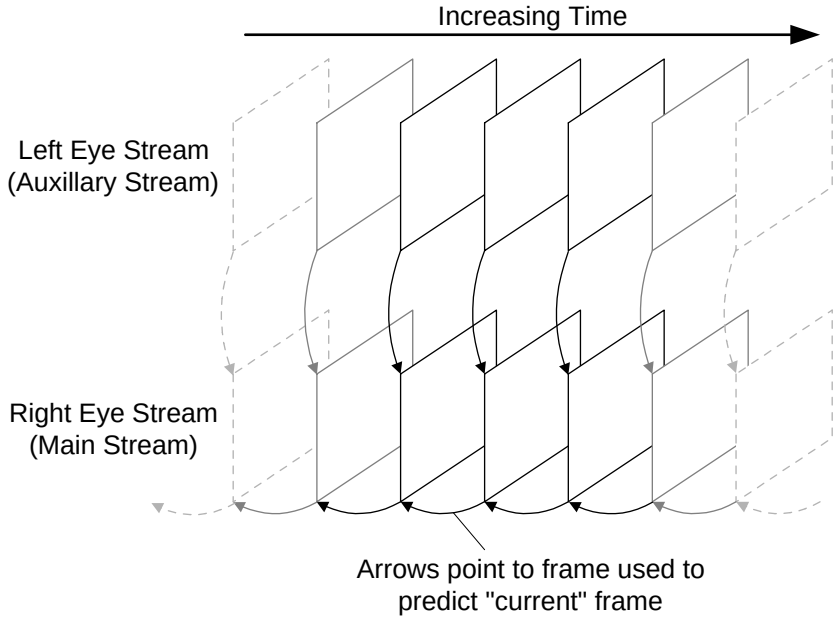


Figure 3.13: Relationships between frames

By taking advantage of certain properties of stereo video, the spatio-temporal algorithm is modified to make it more computationally efficient for disparity estimation. Since the cameras mounted on the platform are assumed to be equipolar, the spatio-temporal algorithm only needs to search the horizontal direction, reducing the number of computations in the full-search and refinement passes. This also implies that the disparity estimation only describes horizontal motion. Thus the number of MV's required to describe the block motion between the current main stream frame and the current auxiliary frame is half the amount that is required to describe the block motion between main stream frames. Another modification is based on the observation that the auxiliary stream always corresponds to the left eye. Because of this, the full-search neighbourhood window for the right image only needs to contain the left side of the reference block position. With respect to the image borders, all objects in the right image are positioned, more to the left than the same objects in the left image.

These two factors reduce the number of computations and are the reasons why the disparity estimation and compensation algorithm will run faster than the general motion estimation algorithm used for the main stream. The output of the disparity estimation and compensation is a set of MV's that describe only horizontal movement and an error residue between the predicted current auxiliary frame and the actual current auxiliary frame. As in the previous section, the horizontal-only MV's can be compressed losslessly using the Lempel-Ziv algorithm and the error residue can be lossy compressed using the foveated SPIHT algorithm.

3.6 Global Motion Compensation

A technique that complements and enhances block-based motion estimation is *global motion estimation* (GME). Motion in a video sequence can be divided into global and local motion. Global motion describes the movement of the recording device usually in terms of pan, tilt, rotation and zoom. Local motion is motion that is not caused by the camera's change in position, such as moving objects in the scene. An advantage of applying GME is an increase in compression, since for scenes containing global motion, a few motion parameters can be transmitted to replace the many locally estimated block motion vectors [49]. A disadvantage is that global motion is difficult to measure from video sequences and increase the already high processing requirements.

In this experiment however, global motion is estimated in a very simple and computationally efficient manner. Since the robotic pivoting platform has encoders attached to the pan and tilt axes, each recorded pair of frames can be associated with a pan and tilt value. These two values can be used to compute a global motion vector. Other simplifications are that the ADS cameras used do not have adjustable zoom and are permanently horizontally mounted to the fixed platform, hence no zooming or rotation is possible. The processing

needed to estimate the global motion is almost zero.

Typically a 6-parameter affined motion model, shown in equation (3.2), is used to describe global motion. The coordinates (x, y) of a pixel in the current image is related to the coordinates (x', y') of the corresponding pixel in the reference image by the parameters m_0, m_1, m_2, m_3, m_4 and m_5. Parameters m_2 and m_5 describe the pan and tilt motion while the remaining parameters describe the rotation and zoom [50].

$$\begin{aligned} x' &= m_0 x + m_1 y + m_2 \\ y' &= m_3 x + m_4 y + m_5 \end{aligned} \tag{3.2}$$

Since the only allowable camera motions are pan and tilt, the 6 parameter model can be simplified to a 2 parameter motion model, shown in equation (3.3), by setting $m_0 = m_4 = 1$ and $m_1 = m_3 = 0$. To obtain the pan and tilt translation parameters, the encoder readings can be converted to translation values using a camera model such as in [51]. However, the camera model presented there is for an extremely high-precision camera rig used to control motion to fractions of a pixel. The camera platform and motion vectors used in this experiment do not have a comparable precision and thus a simpler encoder-to-translation conversion is used. (Recall that the motion vector precision in this implementation is full pixels.)

$$\begin{aligned} x' &= x + m_2 \\ y' &= y + m_5 \end{aligned} \tag{3.3}$$

Two constant conversion factors are used to transform the pan and tilt encoder values to the m_2 and m_5 translation parameters. The constant factors are derived from a feature tracking experiment performed for both the pan and tilt axes. Several features in a video scene are tracked. For each feature, the camera is panned or tilted by some amount and the

ratio of the pixel translation to the encoder counts are manually noted. The experiment is repeated for several features at varying distances from the camera platform and also for different start and stop positions in the frame. The feature tracking experiments showed that overall, the ratios did not vary significantly despite the differences in distance and position in the frame. The final conversion factors (units in pixels/count) for the pan and tilt encoders were calculated by averaging all the experimental ratios and are 1.1 and 0.37 respectively.

3.7 Summary

This section described the goals and design of the stereo video compression algorithm. The design includes components for intraframe, interframe and auxiliary stream compression. The foveated SPIHT algorithm is used for compression of the I-frames and error residues. This algorithm gives the foveated property to the stereo video. A block-based multiresolution spatio-temporal algorithm is used for motion estimation in interframe compression. It combines multiresolution and spatio-temporal techniques with a three-step-search heuristic and a reduced SAD calculation to give a fast algorithm. A slightly modified version of this algorithm is used for the disparity estimation when compressing the auxiliary frames. Small modifications are needed to take advantage of equipolar stereo video properties and make the algorithm more computationally efficient. Global motion derived from encoder readings is also used to increase the compression efficiency.

Chapter 4

Experimental Results

An implementation of the stereo video compression algorithm is evaluated in this chapter. Results for video sequences with different characteristics are presented and analysed qualitatively and quantitatively. The effects of each component in the system is examined with respect to the amount of compression and image quality. Comparisons with the MPEG2 and H.264/MPEG-4 AVC standards will also be performed along with perceptual observations and an analysis of the implementation's computational performance. The complete test videos and resulting compressed versions can be found on the CD-ROM included on the back cover of this thesis.

4.1 Test Videos and Quality Metrics

The stereo video sequences used to test the compression algorithm are captured using the cameras mounted on the robotic panning platform shown previously in Figure 3.1. An application was developed to simultaneously capture stereo frames and encoder readings while the platform orientation was adjusted manually. This allowed the recording of stereo videos that contained different motion characteristics. It is important to test the codec with

different types of motion so that the performance of the algorithm in different situations can be determined. Table 4.1 lists the videos that were generated using the capture program and their corresponding motion characteristics. Each video contains 125 pairs of 8-bit monochrome frames at a resolution of 640×240 pixels along with the corresponding encoder readings.[1] Figures 4.1 through 4.5 show the 60^{th} frame of each of the five videos. The 60^{th} frame is chosen as a reference because, in the following sections, it will not be coded as an I-frame and will be in the middle of a GOF's.

Table 4.1: Test video descriptions

Video Name	Description
Lab1	No object motion, no camera motion. (Still video)
Lab2	Object motion though scene, no camera motion.
Lab3	Object motion at foveation point, no camera motion.
Lab4	No object motion, camera panning.
Lab5	Object motion at foveation point, camera panning. (Tracking moving object)

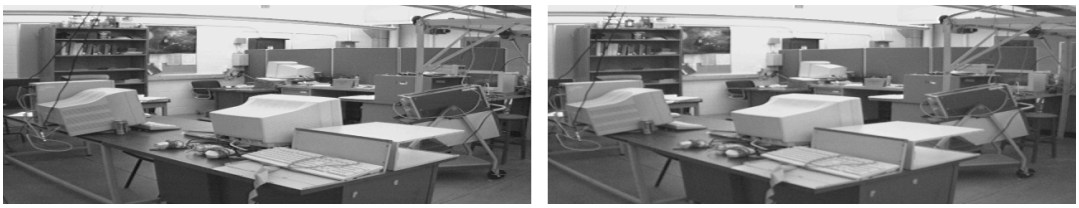


Figure 4.1: Original Lab1 left and right 60^{th} frames

Subjective opinions and quantitative measurements will be used to evaluate the results of the image compression algorithm. Subjective results are generally most useful, although very difficult to quantify without significant user tests. Correlating perceived image quality with an image quality metric is a tricky problem that still does not have a generally

[1]The example video frames appear to be of an odd aspect ratio. This is a result of the VFX3D VR goggles, which require this resolution for the video input. The VR goggle's internally upsample each line to produce video of 640×480 in the display of each eye.

Figure 4.2: Original Lab2 left and right 60^{th} frames

Figure 4.3: Original Lab3 left and right 60^{th} frames

Figure 4.4: Original Lab4 left and right 60^{th} frames

Figure 4.5: Original Lab5 left and right 60^{th} frames

accepted solution [52]. The analysis in this section utilizes two image quality measures to quantitatively evaluate each reconstructed output frame. The first measure is the commonly used peak signal-to-noise ratio (PSNR), shown in equation (4.1). It is measured in units of dB's, with a higher value representing a better match. The second measure is the foveated wavelet quality index (FWQI), shown in equation (4.2), where S($\mathbf{w}$) is the foveated

sensitivity defined in Appendix A, $c(\mathbf{w})$ is the wavelet coefficient value at wavelet location $\mathbf{w}$ and $Q(\mathbf{w})$ is the quality index at wavelet location $\mathbf{w}$.[2] Appendix B derives the FWQI in detail. The FWQI was proposed in [53] for use with foveated images, since it places more emphasis on quality in the foveation region. The metric ranges from 0 to 1, with 1 representing a perfect match.

$$PSNR = 10\, log_{10}\left(\frac{(\text{Max Pixel Magnitude})^2}{\text{Mean Square Error}}\right). \tag{4.1}$$

$$FWQI = \frac{\sum_{\mathbf{w}} S(\mathbf{w})|c(\mathbf{w})|Q(\mathbf{w})}{\sum_{\mathbf{w}} S(\mathbf{w})|c(\mathbf{w})|}. \tag{4.2}$$

4.2 An Incremental Analysis of the Stereo Video Compression Algorithm

This section examines the effects of adding each component in the stereo video compression algorithm separately. Starting from raw video, the rate distortion performance of the algorithm is analysed by first adding intraframe encoding, and then interframe encoding, followed by disparity encoding, and finally global motion compensation. For the stereo test videos of Table 4.1, each uncompressed stereo video has a raw file size of (125 frames/stream)×(2 streams)×(640×240 pixels/frame)×(8 bpp) = 307 200 000 bits = 38 400 000 bytes. Another representation for the video size can be used by fixing the frame rate. If the videos run at 25 fps, the required bitrate for the raw videos will be (2 streams)×(640×240 pixels/frame)×(8 bpp)×(25 fps) = 61 440 000 bps.

[2]The boldface denotes a 2D position vector.

4.2.1 Intraframe Compression Tests

The simplest method to compress video is to compress each frame individually using the foveated SPIHT coder. The statistics for compressing all the videos with independent streams and using only intraframe compression is shown in Table 4.2. The PSNR and FWQI for each frame of the videos is shown in Figures 4.6 and 4.7 respectively. The 60^{th} frame for the videos are shown in Figure 4.8. These tests were based on a SPIHT bitrate of 0.15 bpp, which translates to a final pre-entropy coding size of (640×240 pixels/frame)×(0.15 bpp)×(125 frames/stream)×(2 streams) = 5 760 000 bits or a bitrate of 1 152 000 bps. However, the final file size is smaller after entropy coding as can be seen in the final file sizes of the results. The compression ratios for all the videos are fairly constant since each frame is independently compressed with the same SPIHT bitrate.

Table 4.2: Independent streams, pure I-frame compression results in bytes (0.15 bpp)

Test Video	Main Comp. Size	Aux. Comp. Size	Total Header Size	Final File Size	Final Bitrate (Kbps)	Comp. Ratio	% File Size Decrease Over Raw Video
Lab1	329270	330438	4000	663708	1061.9	57:1	98.3%
Lab2	326895	328233	4000	659128	1054.6	58:1	98.3%
Lab3	328485	328983	4000	661468	1058.3	58:1	98.3%
Lab4	323181	323650	4000	650831	1041.3	59:1	98.3%
Lab5	325202	326120	4000	655322	1048.5	58:1	98.3%

Figures 4.6 and 4.7 seem to show that the auxiliary stream has consistently lower quality for similar compression performance. This offset should not exist if the cameras were perfectly identical. However, although the cameras are the same model, slight differences in the manufacturing and focus may cause one stream to be more conducive for compression and thus create the offset. The cameras are manually subjectively focused and probably do not have the exact same focus setting. Upon close inspection of the original videos, the main stream does have a slightly lower contrast and blurrier image quality when compared

with the auxiliary stream. This will translate into better compression performance since intensity variance is lower and high spatial frequency details are eliminated. This was verified by swapping the video streams before encoding, and checking that the quality metric plots also switched places. The difficulty in calibrating the camera properties is a reason why it is recommended that higher quality cameras be used in future experiments.

It is also interesting to note that the FWQI has a flatter response overall. Its variance over all the frames in each video is more consistent with what is subjectively observed. This is expected since the FWQI places more emphasis on the quality of the foveation region than the PSNR. The foveation region in the reconstructed videos has a more constant quality because the foveated SPIHT algorithm is used to compress the I-frames and residues.

It was discovered during experimentation that applying foveation to stereo compression increases the depth sensation in the foveation region. For low bitrate applications, the sense of object distances in video compressed without foveation is reduced due to the low quality of the entire image. The reduction in detail and loss of edge information in both streams makes the scene lose many depth queues [33]. Higher detail in the foveation region retains more depth cues, resulting in a more accurate depth sensation for objects. Also, important details such as faces, clearly stand out, as opposed to very blocky or blurred faces in unfoveated stereo video.

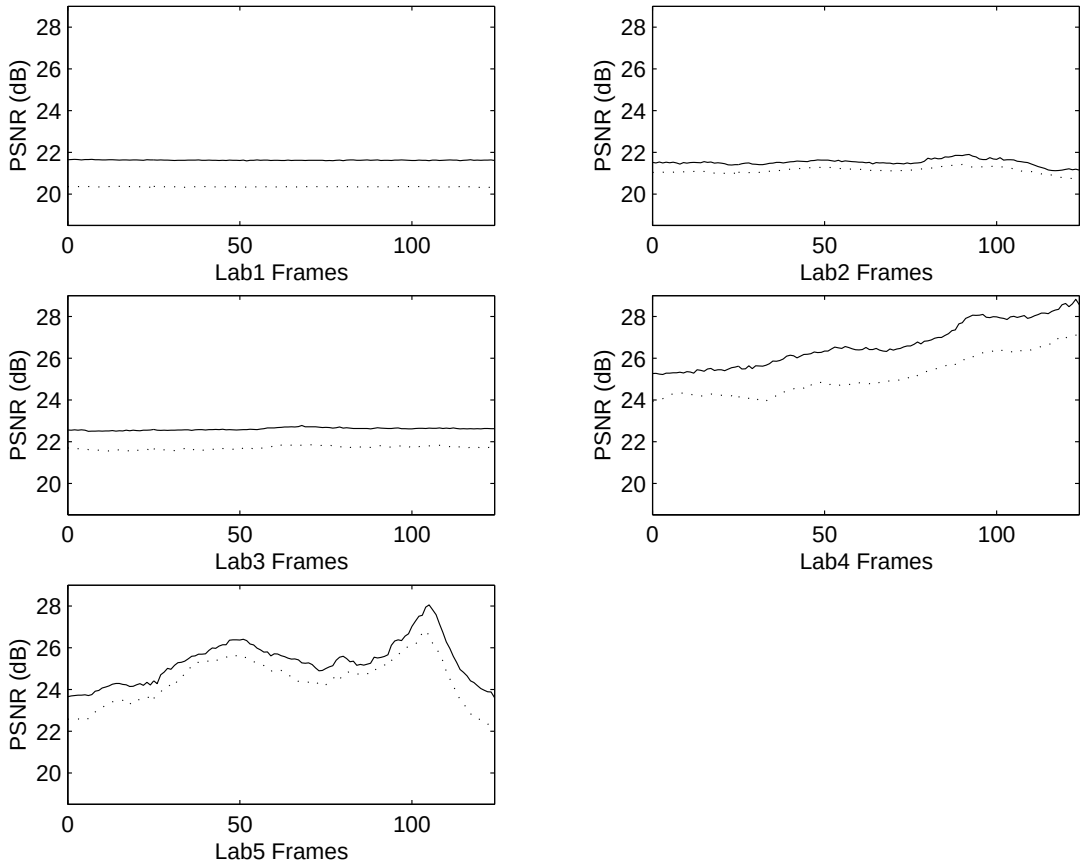


Figure 4.6: PSNR results for intraframe compression (I = 0.15 bpp; Solid line = main stream; Dotted line = auxiliary stream)

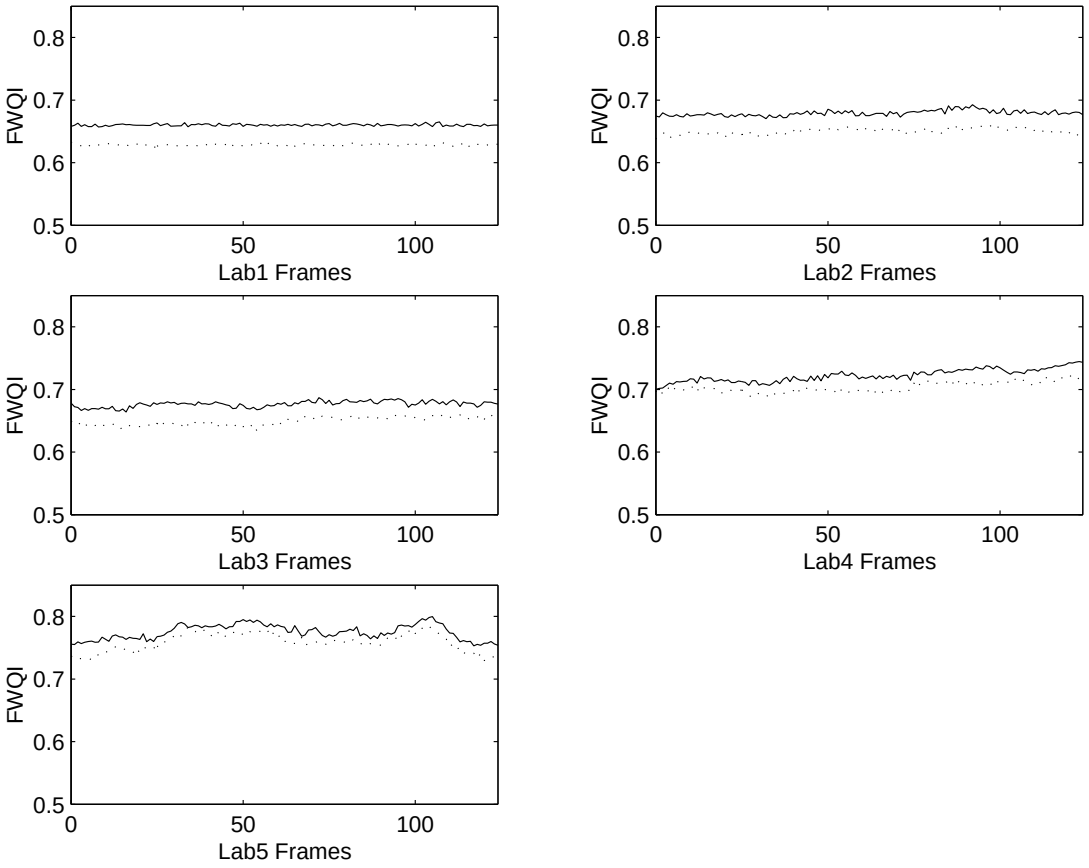


Figure 4.7: FWQI results for intraframe compression (I = 0.15 bpp; Solid line = main stream; Dotted line = auxiliary stream)

Figure 4.8: Intraframe compressed 60^{th} frames (I = 0.15bpp)

4.2.2 Interframe Compression Tests

Applying the fast block-based motion compensation techniques gives the results shown in Table 4.3 and Figures 4.9 and 4.10. The 60^{th} frame for the videos are shown in Figure 4.11. In these tests, both streams were compressed independently of each other, a block

size of 16×16 was used, and the GOF size was 25 (one I-frame every second). The SPIHT algorithm bitrates were 0.15 bpp for the I-frames and 0.03 bpp for the P-frame residues. The search range for the motion estimation was ±15 pixels.

Table 4.3: Independent streams, interframe compression results in bytes (I = 0.15 bpp, P = 0.03 bpp)

Test Video	Main Comp. Size	Aux. Comp. Size	Total Header Size	Final File Size	Final Bitrate (Kbps)	Comp. Ratio	% File Size Decrease Over Intraframe
Lab1	47920	36295	4960	89175	142.7	430:1	86.6%
Lab2	54240	55376	4960	114576	183.3	335:1	82.6%
Lab3	61844	63823	4960	130627	209.0	293:1	80.3%
Lab4	99232	102320	4960	206512	330.4	185:1	68.3%
Lab5	110706	112648	4960	228314	365.3	168:1	65.2%

The results show that interframe compression increases the compression ratio substantially while generally maintaining the video quality for all the test videos. The highest decrease in file size over just intraframe compression was 86.6% for the Lab1 video sequence. Lowest decrease was 65.2% in the Lab5 sequence. The other videos had file size decreases in between those two extreme cases. (Note that the dips occurring at every 25 frames in the PSNR and FWQI plots of Figures 4.9 and 4.10 are caused by the I-frames.)

Lab1 has the highest compression increase since it ideally should have zero motion vectors for all blocks. This translates into very high entropy compression performance when compressing the motion vectors. Occasionally, incorrect matches in the motion estimation will cause motion vectors to be non-zero. These mismatches are mainly in the outer regions of the images because the reconstructed previous frame is used to estimate the motion vectors. Since the outer regions of the reconstructed previous frame are more distorted than the foveation region, more incorrectly matched blocks are expected there.

Lab5 has the lowest increase in compression performance since it has the highest complexity motion of all the sequences. Most of the estimated motion vectors will be non-zero,

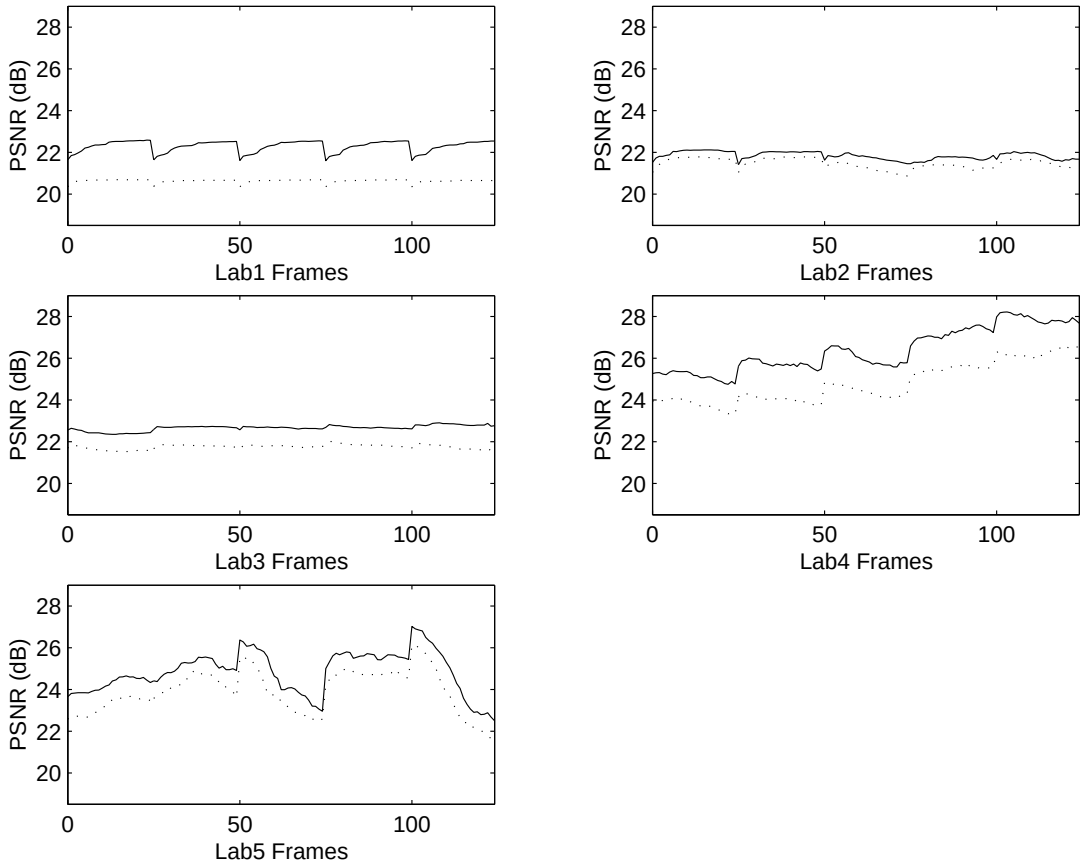


Figure 4.9: PSNR results for independent streams, interframe compression (I = 0.15 bpp; P = 0.03 bpp; Solid line = main stream; Dotted line = auxiliary stream)

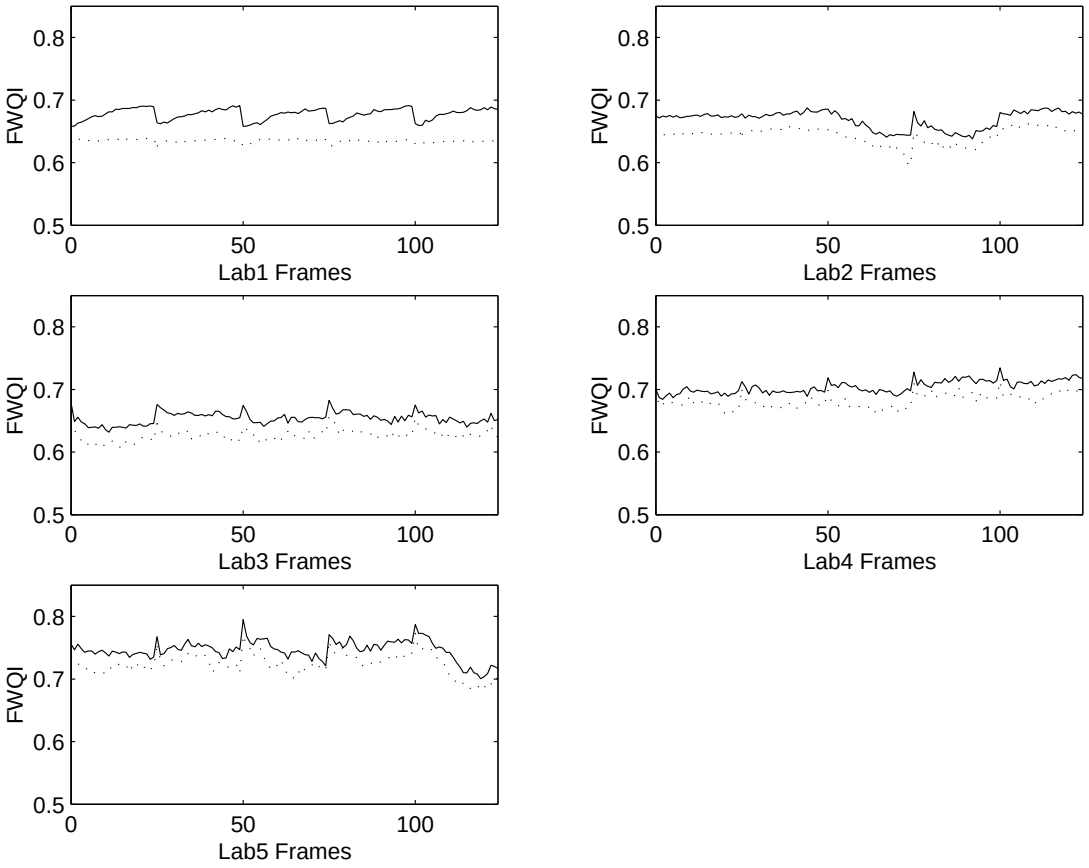


Figure 4.10: FWQI results for independent streams, interframe compression (I = 0.15 bpp; P = 0.03 bpp; Solid line = main stream; Dotted line = auxiliary stream)

Figure 4.11: Interframe compressed 60^{th} frames (I = 0.15bpp, P = 0.03 bpp)

which will not compress as much when applying entropy coding.

At this point, the compression algorithm is most similar to the MPEG2 compression algorithm. Table 4.4 and Figures 4.12 and 4.13 show the results of compressing the test videos using the MPEG2 reference software (TMN5) [54]. Figure 4.14 show the 60^{th} frames

compressed using MPEG2. The values in Table 4.4 are slightly different from the tables in the rest of this section because they represent the file sizes including header information. The interframe compression algorithm and the MPEG2 software produce similar compressed file sizes.

Table 4.4: Independent streams, MPEG2 compression results in bytes

Test Video	Main Comp. File Size	Aux. Comp. File Size	Total File Size	Final Bitrate (Kbps)	Comp. Ratio
Lab1	44488	45413	89901	143.8	427:1
Lab2	56964	57175	114139	182.6	336:1
Lab3	52572	54529	107101	117.4	358:1
Lab4	96992	102566	199558	319.3	192:1
Lab5	114159	115279	229438	367.1	167:1

The PNSR results suggest that the quality of the MPEG2 compressed videos are higher than those compressed by the algorithm presented thus far. This is a false indication, because the MPEG2 videos are not foveated. A better comparison is to instead use the FWQI to evaluate the results. The FWQI shows that the foveated compression algorithm presented here has a similar or better foveal quality than the MPEG2 compressed video for a similar bitrate. Subjectively, the MPEG2 video is much blockier than the proposed algorithm. The next sections add disparity compensation and global motion compensation to the algorithm, causing the overall compression performance to be better than MPEG2.

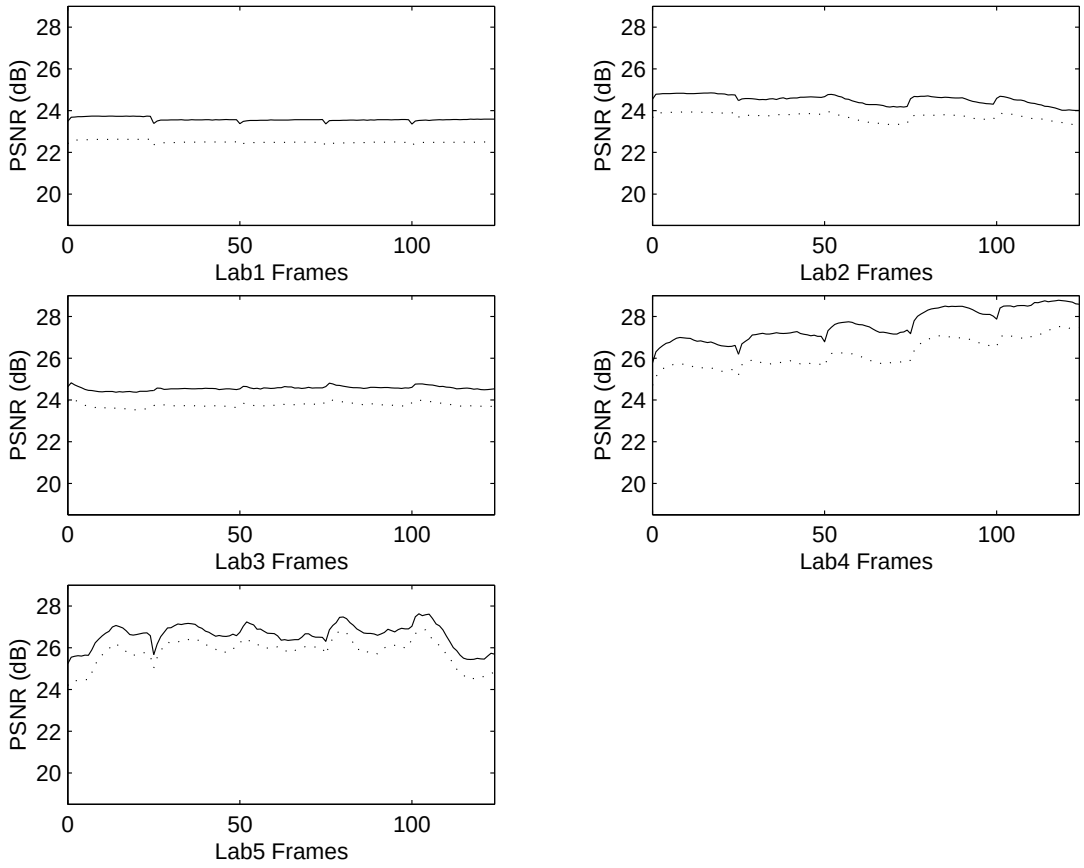


Figure 4.12: PSNR results for independent streams, MPEG2 compression (Solid line = main stream; Dotted line = auxiliary stream)

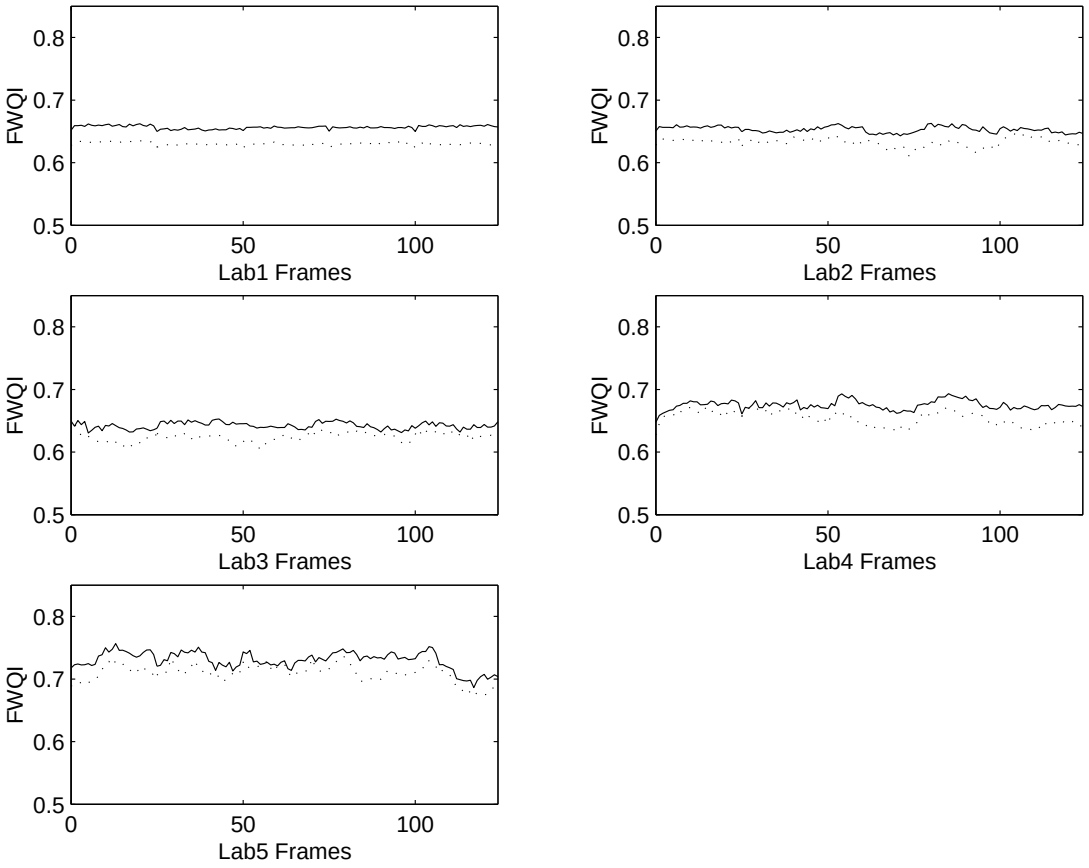


Figure 4.13: FWQI results for independent streams, MPEG2 compression (Solid line = main stream; Dotted line = auxiliary stream)

Figure 4.14: MPEG2 compressed 60^{th} frames

4.2.3 Auxillary Stream Compression Tests

The effect of compressing the auxiliary stream using the disparity compensation are examined in this section. The main stream remains independently compressed using the same settings as in the previous section, but now the auxiliary stream is reconstructed from the

main stream as described in Section 3.5. The S-frame residues are compressed using the same SPIHT bitrate of 0.03 bpp and the search range for the disparity estimation is +31 pixels. A larger search range is required since the matching feature in the main stream frame is generally at a greater distance than if the feature was matched from the previous auxiliary frame. Table 4.5 and Figures 4.15 and 4.16 show the results of these tests on the five video sequences. The 60^{th} frame for the videos are shown in Figure 4.17.

Table 4.5: Stereo interframe compression results in bytes (I = 0.15 bpp, P = 0.03 bpp, S = 0.03 bpp)

Test Video	Main Comp. Size	Aux. Comp. Size	Total Header Size	Final File Size	Final Bitrate (Kbps)	Comp. Ratio	% File Size Decrease Over Interframe
Lab1	47920	23712	4980	76612	122.6	501:1	14.1%
Lab2	54240	27976	4980	87196	139.5	440:1	23.9%
Lab3	61844	30936	4980	97760	156.4	392:1	25.2%
Lab4	99232	31349	4980	135561	216.9	283:1	34.4%
Lab5	110706	44782	4980	160468	256.7	239:1	29.7%

Applying disparity compensation coding causes the auxiliary stream compressed size to decrease. The main stream compression performance remains the same since it is coded independently. In these tests, Lab4 had the greatest decrease in file size over interframe compression of 34.4%. Lab1 had the lowest decrease of 14.1%. This is explained by the fact that Lab1 already had very high compression due to its mainly zero motion vectors. The disparity motion vectors, however, will not be zero since almost all the blocks in the auxiliary stream will be displaced by some amount with respect to the main stream. Lab5's auxiliary stream compression performance increases since the disparity motion vector variance is much lower than the original interframe motion vector variance. This results in more efficient entropy coding. Also recall that the vertical component of the motion vectors is not used, due to the equipolar assumption, futher reducing the auxiliary stream file size.

A problem visible subjectively and also quantitatively in Figures 4.15 and 4.16, is that

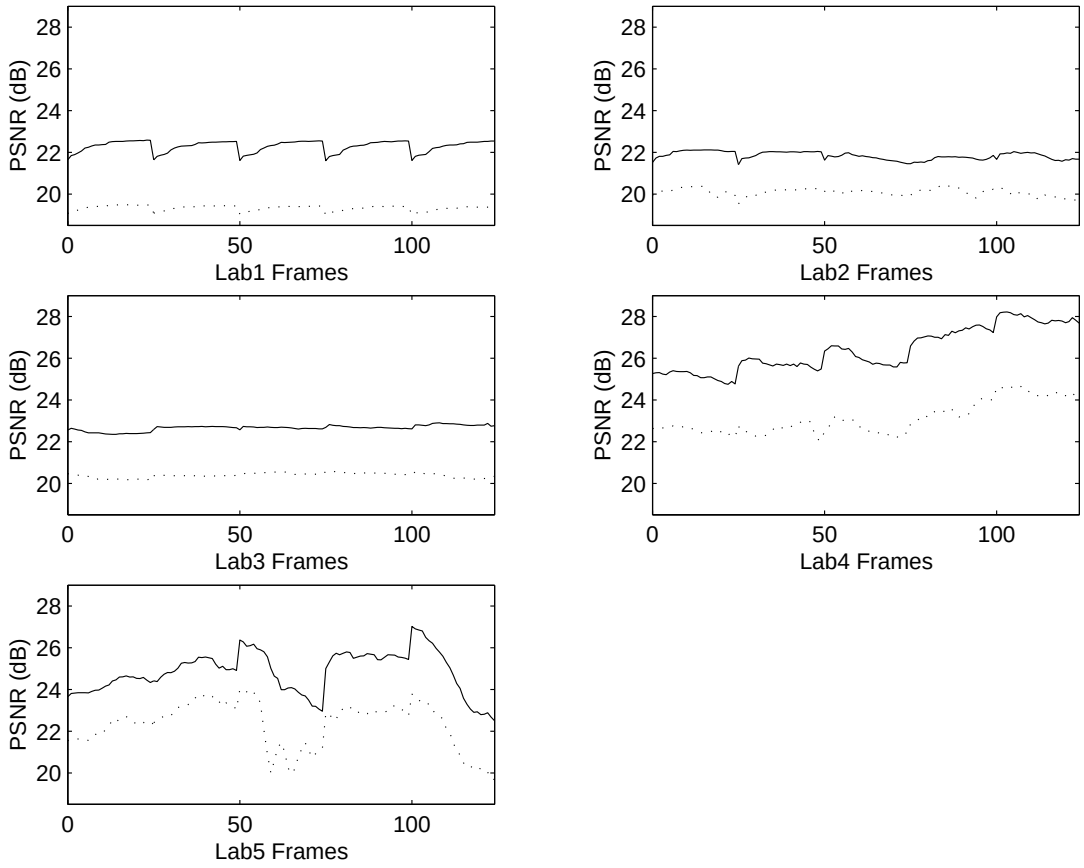


Figure 4.15: PSNR results for stereo interframe compression (I = 0.15 bpp; P = 0.03 bpp; S = 0.03 bpp; Solid line = main stream; Dotted line = auxiliary stream)

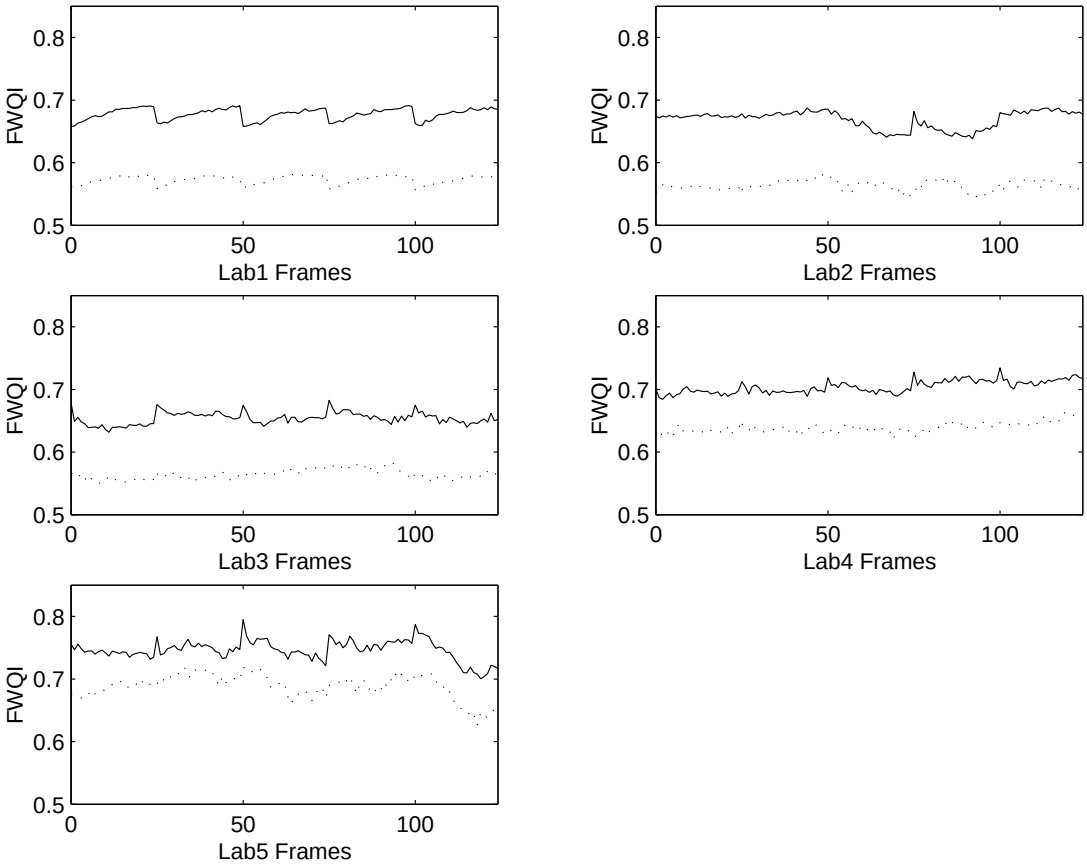


Figure 4.16: FWQI results for stereo interframe compression (I = 0.15 bpp; P = 0.03 bpp; S = 0.03 bpp; Solid line = main stream; Dotted line = auxiliary stream)

Figure 4.17: Stereo compressed 60^{th} frames (I = 0.15bpp, P = 0.03 bpp, S = 0.03 bpp)

the video quality of the auxiliary stream has substantially reduced. There are three causes of this degradation in video quality. Disparity compensation is inherently less accurate than interframe compensation. This is because there are more occluded regions between corresponding main and auxiliary frames than there are between frames of the same stream.

The second reason is that the ADS cameras have high lens distortion especially at the border regions. The disparity estimation algorithm's assumption that the frames are equipolar is now false at the left and right sides of the frames. Figure 4.18 points out a region where the equipolar assumption is false. This causes more incorrect block matches at the sides of the video frames. Cameras with less lens distortion will solve this problem, or the equipolar assumption will have to be relaxed. Allowing the disparity compensation to have small vertical motion components will solve many mismatches caused by the lens distortion. However, this will increase the compressed file size.

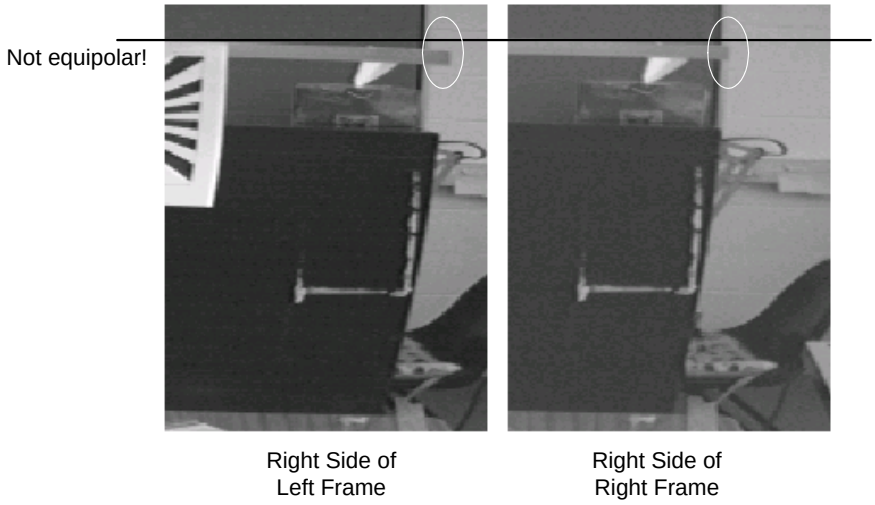


Figure 4.18: Example region where equipolar assumption is false

The final reason for the drop in PSNR and FWQI is because the original main and auxiliary frames were recorded from different camera sources. As mentioned previously, the cameras generate different video characteristics, such as focus, contrast, etc. At the decoder, the auxiliary stream is reconstructed from the main stream and therefore its video characteristics will be closer to the main stream characteristics. This causes the PSNR and FWQI calculations to report a lower correlation with respect to the original auxiliary

stream.

Of the three degradation causes, only the first two are subjectively noticeable. However, the last cause is the main source of the drop in PSNR and FWQI since it affects whole frames and not just a few isolated blocks. To reduce the noticeable artifacts, the S-frame residue bitrate can be increased. Increasing the S-frame residue SPIHT bitrate to 0.045 bpp results in Table 4.6 and Figures 4.19 and 4.20. Notice that the file size for the auxiliary stream is now much higher than when the S-frame residues were coded using 0.03 bpp. However, subjectively and quantitatively, the increase in bitrate does not help the video quality enough to justify the resulting file size.

Although the quality of the reconstructed auxiliary stream is poor when viewed alone, the noticeable artifacts are less prominent when viewed stereoscopically with the main stream. The suppression theory of binocular vision explains this phenomenon [35]. It may be possible to reduce the quality (and increase compression) of the auxiliary stream even more without users realizing the degradation. The exact amount of degradation in relation to this experiment should be examined in future psychological experiments.

Table 4.6: Stereo interframe compression results in bytes (I = 0.15 bpp, P = 0.03 bpp, S = 0.045 bpp)

Test Video	Main Comp. Size	Aux. Comp. Size	Total Header Size	Final File Size	Final Bitrate (Kbps)	Comp. Ratio	% File Size Decrease Over Interframe
Lab1	47920	44831	4980	97731	156.4	392:1	9.6%
Lab2	54240	49943	4980	109163	174.7	351:1	4.7%
Lab3	61844	57705	4980	124529	199.2	308:1	4.7%
Lab4	99232	53190	4980	157402	251.8	243:1	23.8%
Lab5	110706	70916	4980	186602	298.6	205:1	18.3%

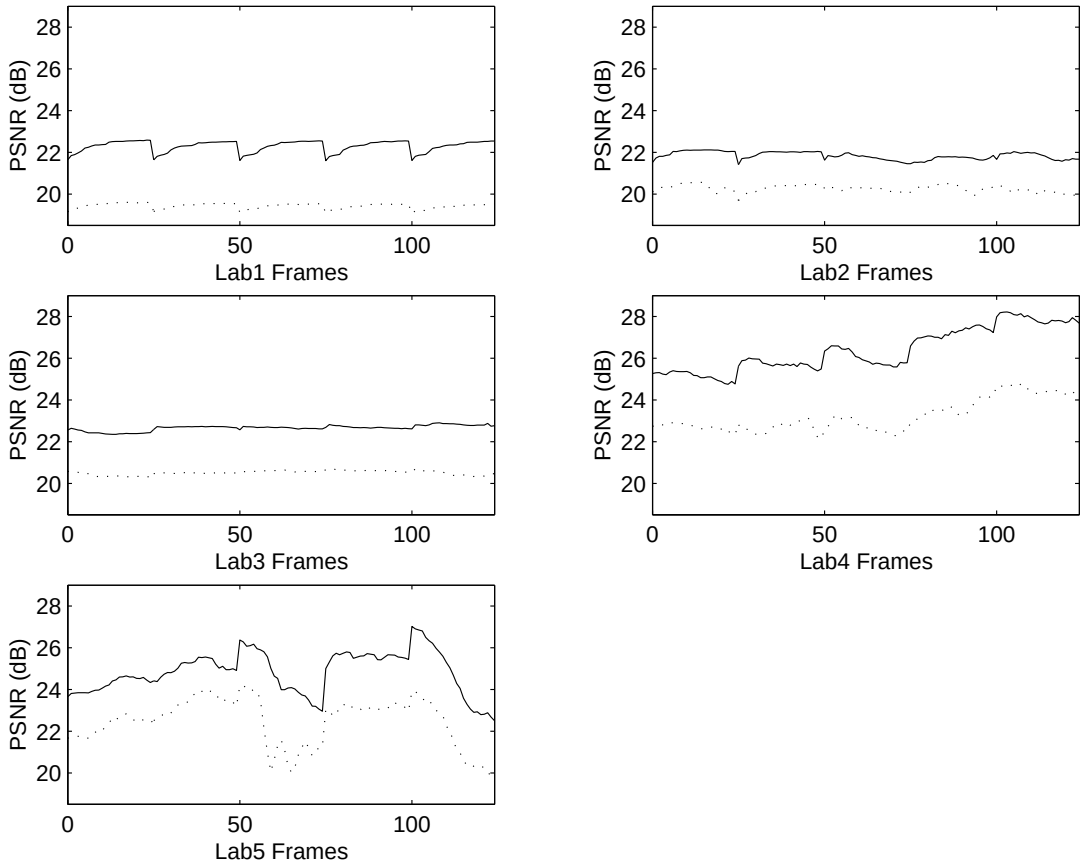


Figure 4.19: PSNR results for stereo interframe compression (I = 0.15 bpp; P = 0.03 bpp; S = 0.045 bpp; Solid line = main stream; Dotted line = auxiliary stream)

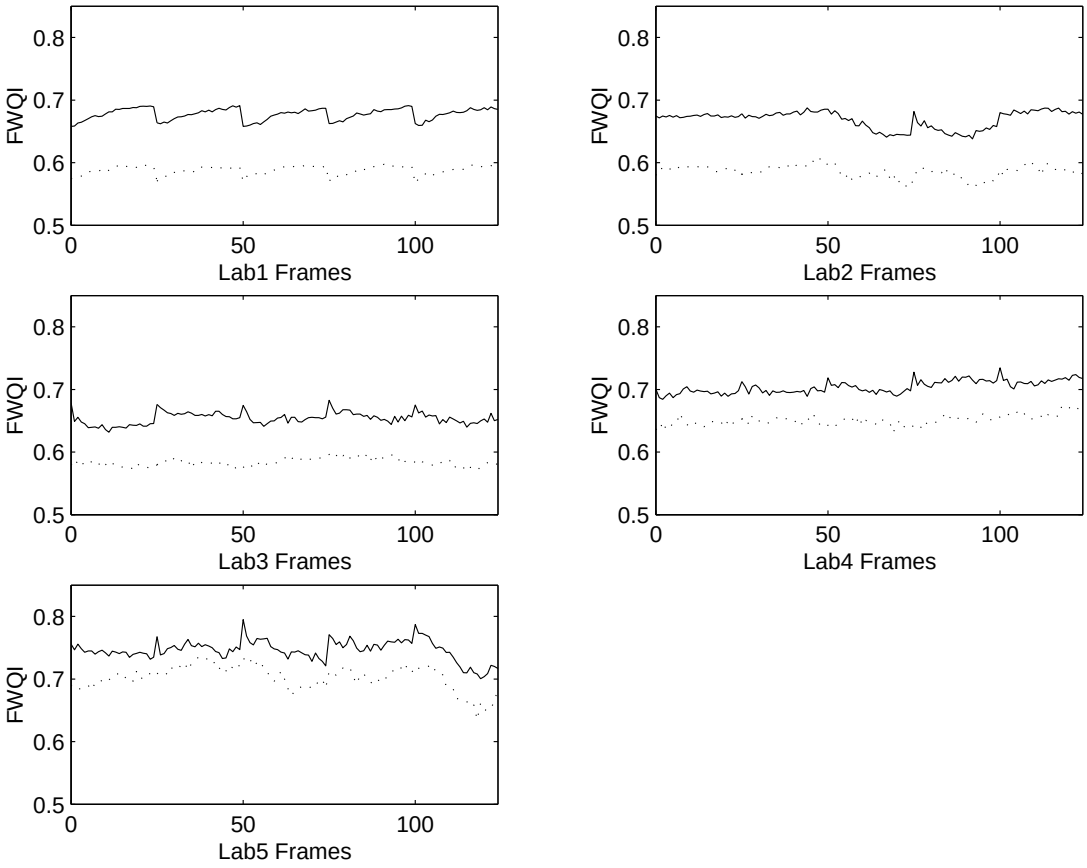


Figure 4.20: FWQI results for stereo interframe compression (I = 0.15 bpp; P = 0.03 bpp; S = 0.045 bpp; Solid line = main stream; Dotted line = auxiliary stream)

4.2.4 Global Motion Compensation Tests

Global motion compensation (GMC) is enabled for the tests in this section. All the settings are the same as the first set of tests in the previous section. As verification for the simple global motion calculation, the global motion vectors generated by the encoders are compared to motion vectors generated by the global motion estimation algorithm in the Joint Video Team's (JVT) H.264/MPEG-4 AVC reference software [55]. The H.264/MPEG-4 AVC algorithm only uses the video to estimate the global motion. Figures 4.21 and 4.22 show the computed and estimated horizontal motion for the Lab4 and Lab5 video sequences. Vertical motion is not shown since it is minimal in the videos and its computation almost identical. Only the motion vectors estimated from the right eye video stream are shown since the motion vectors estimated from the left eye are similar.

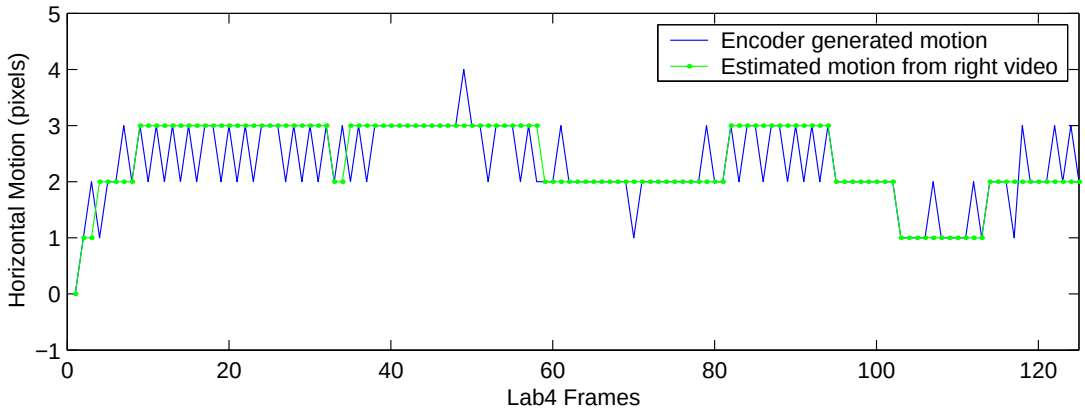


Figure 4.21: Comparison of horizontal global motion for Lab4

The results indicate that the encoder generated motion matches closely to the motion estimated by the H.264/MPEG-4 AVC software. The differences are usually ±1 pixel, except in one frame where the difference is 2 pixels. Since the encoder readings represent the actual recorded global motion, the calculated global motion should be more accurate than the H.264/MPEG-4 AVC estimated motion, which can explain the small differences

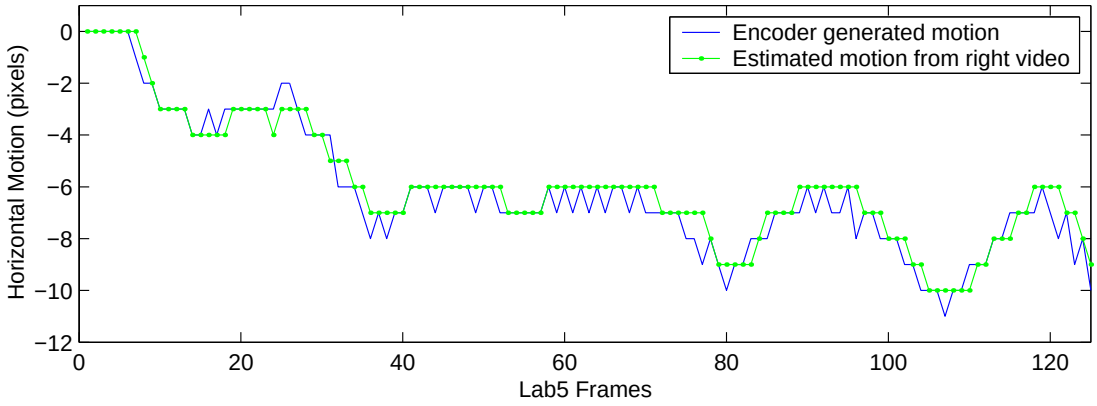


Figure 4.22: Comparison of horizontal global motion for Lab5

between the results. The close motion vector matching of the two methods, suggest that the motion vectors generated from the simple encoder reading conversion are correct, and can be used in place of a complex global motion estimation algorithm.

The video encoding results of applying GMC are shown in Table 4.7 and Figures 4.23 and 4.24.

Table 4.7: Global motion compensated stereo interframe compression results in bytes (I = 0.15 bpp, P = 0.03 bpp, S = 0.03 bpp)

Test Video	Main Comp. Size	Aux. Comp. Size	Total Header Size	Final File Size	Final Bitrate (Kbps)	Comp. Ratio	% File Size Decrease Over Stereo Comp.
Lab1	47789	23733	5230	76752	122.8	500:1	-0.1%
Lab2	55613	28202	5230	89045	142.5	431:1	-2.1%
Lab3	61761	30840	5230	97831	156.5	392:1	-0.1%
Lab4	72525	31892	5230	109647	175.4	350:1	19.1%
Lab5	98824	44710	5230	148764	238.0	258:1	7.9%

The results show that using GMC, the Lab4 and Lab5 video sequences had, respectively, a 19.1% and 7.9% decrease in file size over normal stereo compression while maintaining similar quality. These are the only sequences that contained global motion and hence the increase in compression performance additionally justifies that the simple conversion factor

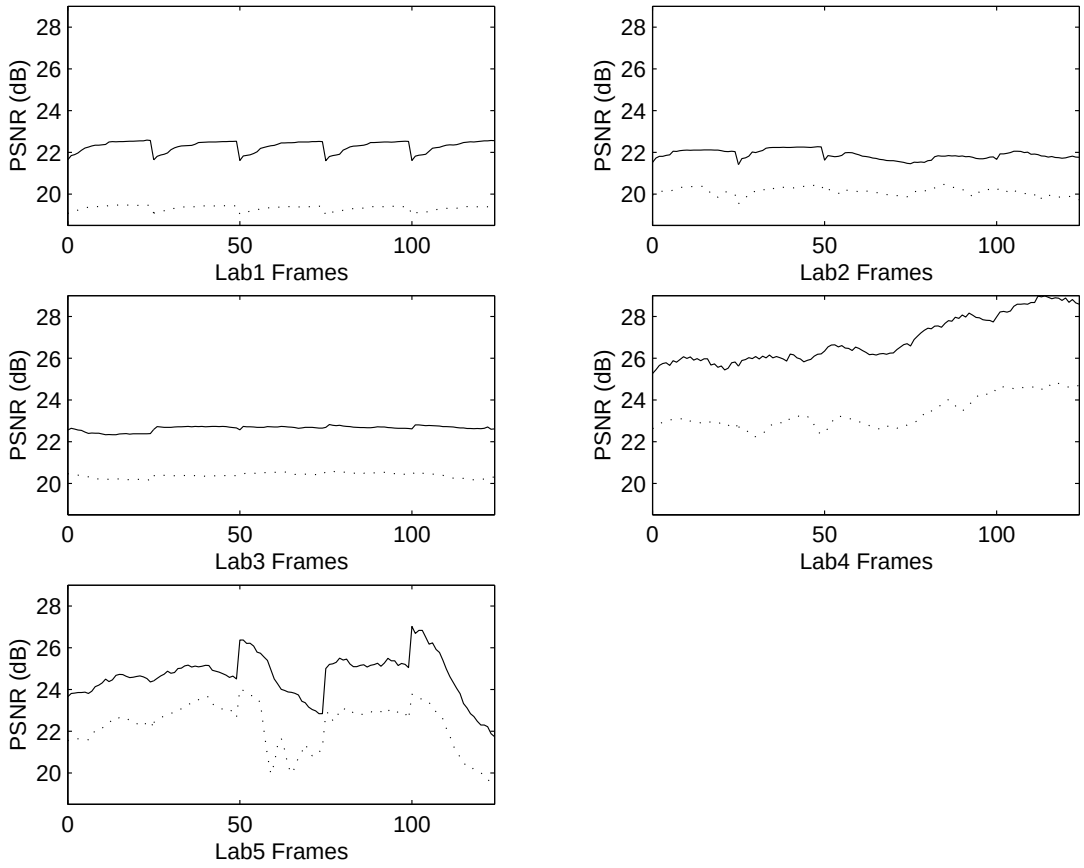


Figure 4.23: PSNR results for GMC stereo interframe compression (I = 0.15 bpp; P = 0.03 bpp; S = 0.03 bpp; Solid line = main stream; Dotted line = auxiliary stream)

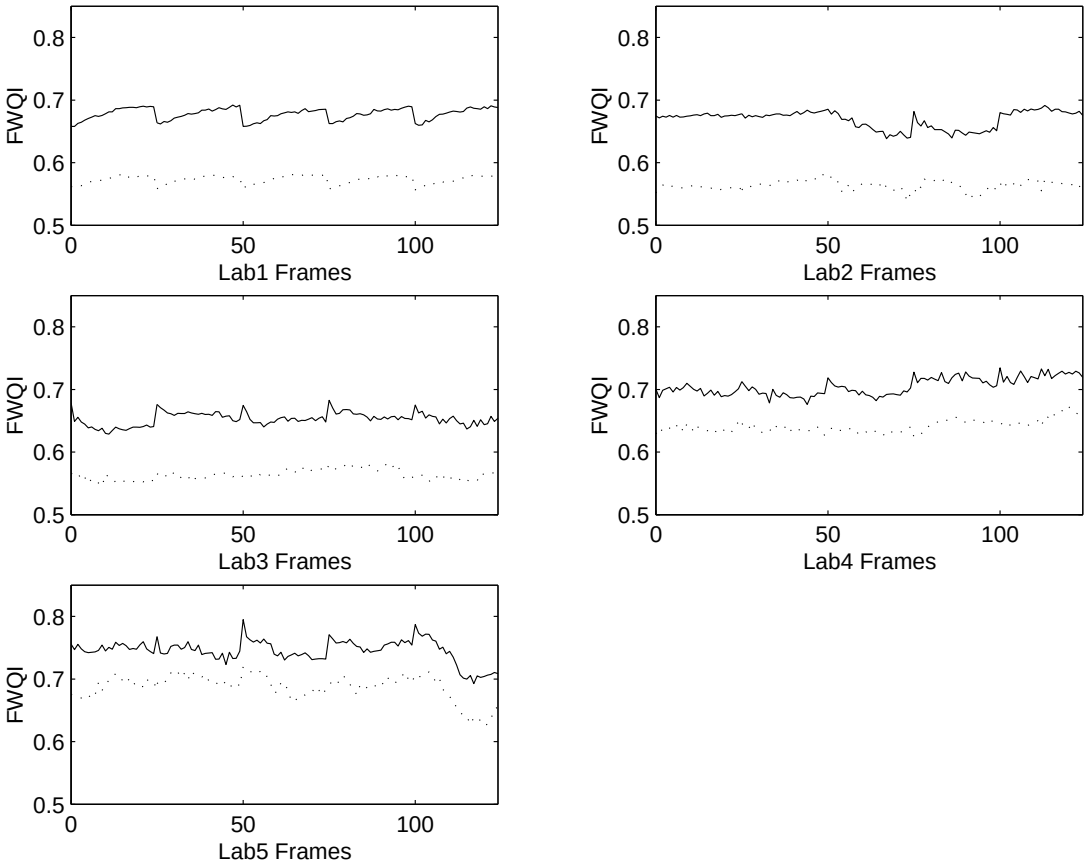


Figure 4.24: FWQI results for GMC stereo interframe compression (I = 0.15 bpp; P = 0.03 bpp; S = 0.03 bpp; Solid line = main stream; Dotted line = auxiliary stream)

method used to calculate the global motion vector from the encoder readings, is adequate for GMC. The improvement for Lab5 is not as great as for Lab4, because there is a high amount of local motion in the lab5 sequence.

The Lab1, Lab2 and Lab3 sequences, which do not have any global motion, did not gain additional compression but instead had slightly larger final file sizes. The increase in file size can be traced to the additional header data that is used to transmit the global motion vector of each frame.

The global motion compensated stereo compressed videos are now compared with videos compressed using the H.264/MPEG-4 AVC encoder. Although this is not exactly a fair comparison (since the H.264/MPEG-4 AVC codec contains many other features, such as intra prediction, $\frac{1}{4}$-pixel motion compensation, variable-size blocks, etc. that make it superior [56]), it will be interesting to see where the stereo compression algorithm stands in relation. The H.264/MPEG-4 AVC coder is set to have a GOF of 25 and set for maximum compression with GMC on.[3] The results are shown in Table 4.8 and Figures 4.25 and 4.26.

Table 4.8: Independent streams, H.264/MPEG-4 AVC reference software compression results in bytes

Test Video	Main Comp. File Size	Aux. Comp. File Size	Total File Size	Final Bitrate (Kbps)	Comp. Ratio
Lab1	16895	19550	36445	58.3	1053:1
Lab2	23436	25524	48960	78.3	784:1
Lab3	22717	25670	48387	77.4	793:1
Lab4	16183	19960	36143	57.8	1062:1
Lab5	40479	42281	82760	132.4	464:1

From the results, it is clear that even though the main and auxiliary streams are compressed individually, the H.264/MPEG-4 AVC codec outperforms the stereo video compres-

[3]The H.264/MPEG-4 AVC coder does not explicity have a compression size setting. The quantisation value (0-31) selects the quality of the reconstructed video and hence affects the compression rate. For maximum compression a quantisation value of 31 is used. [55]

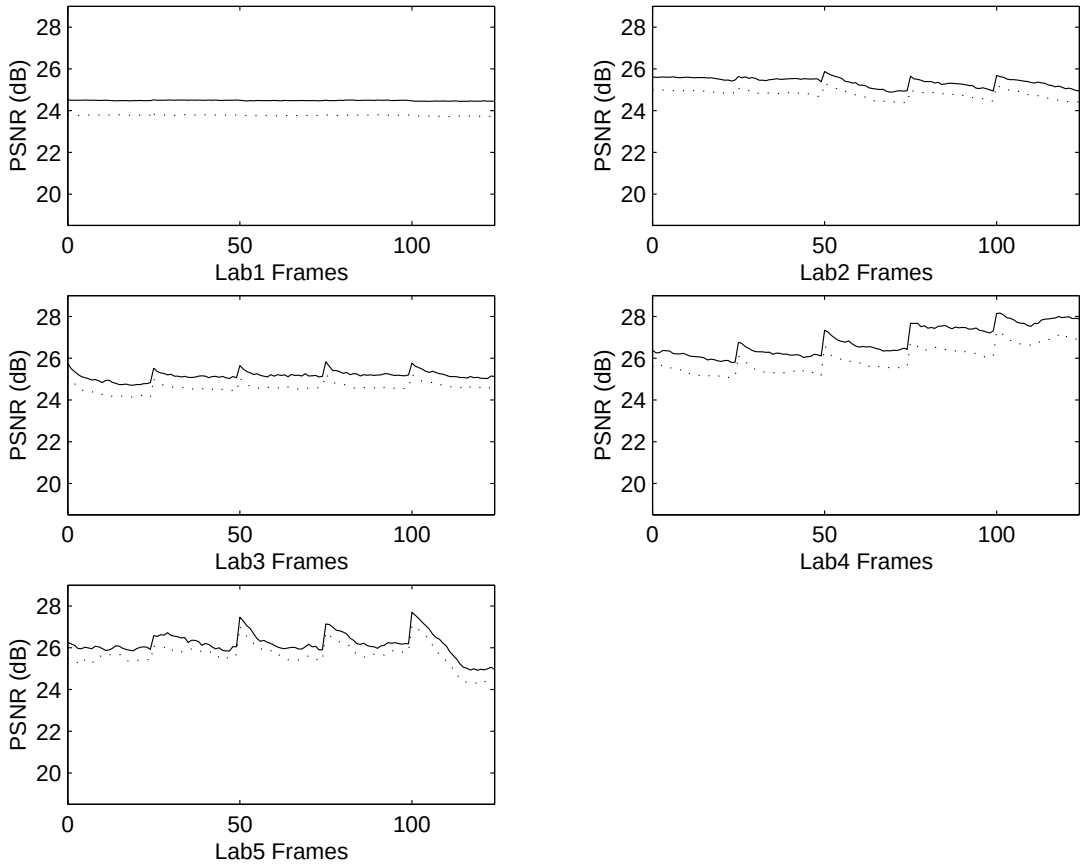


Figure 4.25: PSNR results for independent streams, H.264/MPEG-4 AVC compression (Solid line = main stream; Dotted line = auxiliary stream)

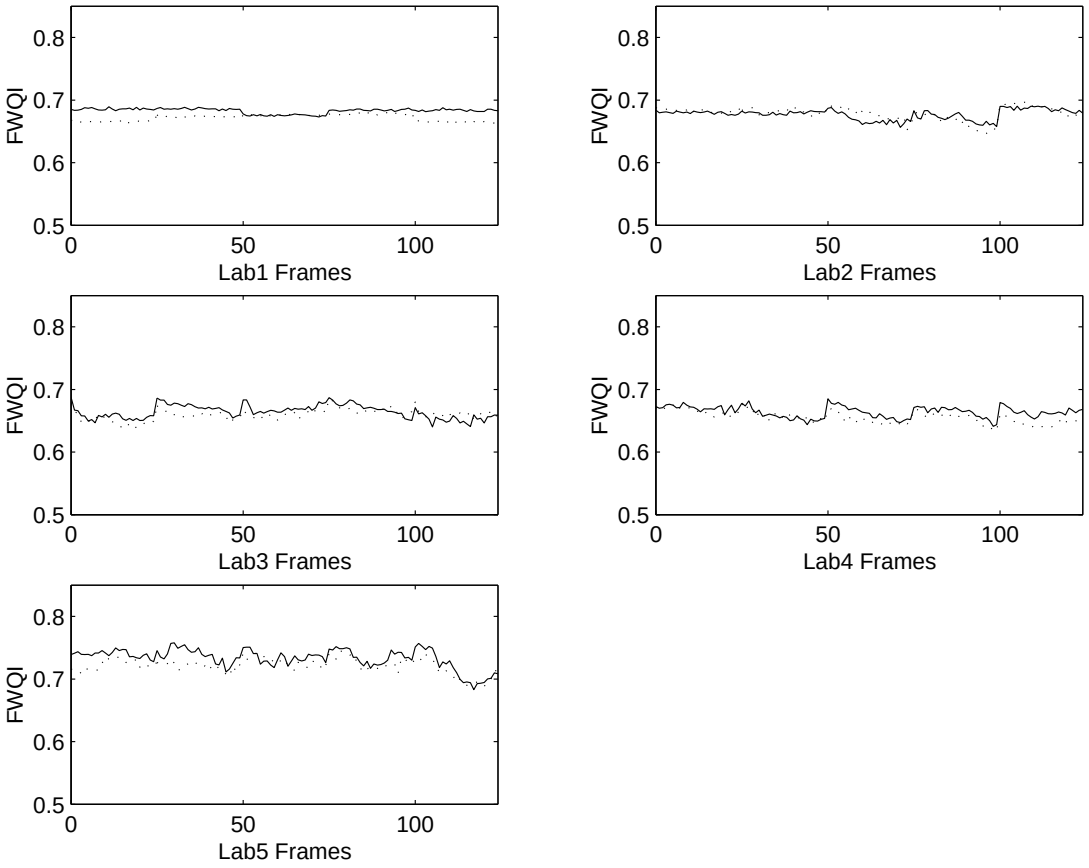


Figure 4.26: FWQI results for independent streams, H.264/MPEG-4 AVC compression (Solid line = main stream; Dotted line = auxiliary stream)

sion algorithm in terms of compression rate and quality. However, the previous sections have shown that all the components in the stereo compression system contribute greatly to reducing the compressed video file size and computation time. Combining the above techniques into the H.264/MPEG-4 AVC codec will create a faster more highly compressed foveated stereo video algorithm. Future research should focus on implementing the stereo video algorithm features into the H.264/MPEG-4 AVC reference software framework.

As stated in the Chapter 3, the computers used in this experiment were Intel Pentium III 866MHz personal computers. The implementation of the algorithm was done completely in software (C++) using no processor dependent optimisations and was programmed for correctness instead of speed. Although the result was that the implementation did not meet the real-time processing requirement of 25 fps stereo video, the implementation was still considerably faster than the H.264/MPEG-4 AVC reference software. The H.264/MPEG-4 AVC reference software encoder took on average 2500 s to encode each stereo video sequence, in comparison to the average encoding time of 106 s that the implemented stereo video encoder required. In other words, the implemented stereo video encoder is about 24 times faster than the H.264/MPEG-4 AVC reference software. The majority of the encoding processing time was taken by the block-based motion estimation algorithm and the DWT implementation. (The DWT implementation used the lifting algorithm as described in [57].)

Even though this implementation did not meet the real-time processing requirements of 25 fps stereo video, the algorithm does not contain any intrinsic delays, so given a faster processor, dedicated hardware or more efficient software implementation, the design will be able to run at 25 fps. It is the opinion of the author that a real-time 25 fps software only implementation is achievable, if processor dependent optimisations, such as the Intel MMX extensions, along with a more streamlined implementation are used.

4.3 Summary

This chapter showed results of the stereo video compression algorithm in a step-by-step manner. The results showed that each component contributed greatly to the overall compression ratio. The algorithm produces compressed video that is competitive with MPEG2, but is outperformed by H.264/MPEG-4 AVC. Incorporating aspects of this algorithm into the H.264/MPEG-4 AVC framework should bring the compression ratio and quality to a similar level. Perceptual observations showed that the foveation added greatly to the depth sensation in the foveal region. The final result is a stereo video compression algorithm for visual telepresence that incorporates foveated intraframe coding, block-based interframe and disparity coding, and global motion compensation. Even though the algorithm could not run in real-time on the experiment computers, the restriction was in the computation power and software implementation. Given a fast enough processor or better software optimisation, a real-time 25 fps implementation is possible.

Chapter 5

Conclusions

The goal of this thesis was to combine foveation into the design of a stereo video compression algorithm for visual telepresence applications. It was also desired that aspects of a telepresence system, such as the camera platform's encoders, be considered in the algorithm. Because of the application, the algorithm could not have any inherent delays and a compressed bitrate significanlty lower than 10 Mbps was the goal. The main components in the compression system were:

- Foveated SPIHT image compression.
- Fast block-based interframe compression.
- Fast block-based disparity compression.
- Global motion compensation.

The foveated SPIHT compression algorithm was the mechanism to foveate each frame of video. It had a high compression rate and produced images that were pixel accurate psychologically foveated. The compressed file size is easily adjustable with this algorithm since the bitrate can be specified explicitly.

The spatio-temporal fast block-based motion estimation algorithm was used for the interframe and disparity compression. It provided a fast algorithm that generated motion vectors for each block in the main and auxiliary frames. The compression ratio was greatly increased by applying this algorithm to both motion estimation components. However, the disparity compensation produced lower quality video for the auxiliary stream. The three causes identified were:

- Disparity compensation poorer in general.
- Camera lenses introduced image distortion.
- Difficulty adjusting cameras to same video properties.

The last two causes can be solved by using higher quality cameras, or allowing the disparity compensation motion vectors to have small vertical motion components. This means that the equipolar assumption would have to be relaxed.

Motion of the camera platform was recorded for each frame of video. A simple conversion factor calculated the global motion vectors from the encoder readings. This method proved to closely match the motion estimated by the much more complex and time consuming H.264/MPEG-4 AVC. GMC only improved the compression of test videos that contained global motion.

Together, these components formed the complete foveated stereo video compression algorithm. Comparisons were made to the MPEG2 and H.264/MPEG-4 AVC standards. This algorithm had better compression performance than MPEG2 because of the disparity compensation and GMC, but was inferior to H.264/MPEG-4 AVC. However, the implemented algorithm encoder was 24 times faster than the H.264/MPEG-4 AVC reference software encoder.

5.1 Contributions

This work was an extension of the work done by Turner in [11]. The original goal of this algorithm was to simply exceed his compression factor while maintaining as much video quality as possible. The results here have shown that it is possible to surpass his compression factor by at least 10 times.

The combination of the compression algorithm components into a complete working system is also another contribution. This thesis has shown that the foveated SPIHT algorithm can be used as the interframe and residue coder to generate a foveated video stream.

It was also shown that a simple conversion factor can be used to accurately compute the full-pixel global motion vectors from encoder readings. Visual telepresence systems should always use this fact to replace time-consuming global motion estimation algorithms.

5.2 Future Work

Before any new research is performed, the cameras on the robotic platform should be replaced with ones of much higher quality and more easily configurable. As can be seen in the previous chapter, the lens distortion and the difference between camera properties in the current cameras is extreme. This makes processing more difficult causing more mismatched blocks. Higher quality stereo videos will have fewer mismatched blocks and hence would be more compressible.

Also, serious thought needs to be put into how to mount the cameras on the platform in a manner that allows easy equipolar alignment. The camera mounts should also have the ability to lock down the position of the cameras once they have been calibrated.

Futher research in this area should focus on incorporating the concepts of this thesis with advanced standards, such as H.264/MPEG-4 AVC. The many features already developed in H.264/MPEG-4 AVC can be used to increase the performance of the current system. It

is recommended that the actual software implementation of the stereo video compression algorithm be abandoned, and instead, features of it be incorporated into the H.264/MPEG-4 AVC reference software [55] framework.

Other methods for foveation should be explored, especially the algorithms proposed by Geisler and Perry in [58]. Their methods reduce the amount of original data by constructing a multiresolution pyramid. They claim that their methods can be interfaced with other video compression algorithms producing foveated results that are compressed 2-3 times smaller in size.

More research needs to be performed on increasing the speed of the algorithms. Processor dependent optimisations, dedicated hardware and other fast algorithms should be examined for future implementations. Algorithm speed is currently the only limiting factor to developing a true real-time stereo video compression algorithm. The foveated stereo compression algorithm also needs to be incorporated into a complete working visual telepresence system so that psychological user experiments can be performed and analysed.

Appendix A

Foveation Sensitivity Mask Calculation

This appendix derives the normalised sensitivity function used to calculate foveation weighting masks for the Foveated SPIHT algorithm. The derivation begins from a mathematical model that fits the contrast sensitivity (sensitivity to detail), CS, of a typical retina, equation (A.1). $CS(f, e)$ is a function of the eccentricity e and spatial frequency f [40]. Spatial frequency, measured in cycles/degree, is the inverse of the periodicity with which the image intensity values change. Image features that have a greatly varying intensity over a short distance (e.g. edges) have high spatial frequency. Retinal eccentricity, measured in degrees, is the angular position in degrees from the centre of the retina (fovea). At 0 degrees the eye has the highest concentration of cones and therefore detects more details of the scene. The concentration of cones decreases exponentially as the eccentricity increases.

$$CS(f, e) = \frac{64}{exp\left(0.106 f \frac{e+2.3}{2.3}\right)}. \tag{A.1}$$

Equation (A.1) is the starting point used to derive the wavelet domain error sensitivity

function, which calculates a normalised error sensitivity given the position $\mathbf{w}$[1] of a DWT coefficient. The computation of the wavelet domain error sensitivity function takes two steps. The first step is to compute a space-variant sensitivity function. The second step is to compute the general sensitivity of each wavelet subband since the visual significance of each subband differs [40]. Combining the two sensitivities will give the final wavelet domain sensitivity function.

Step one is to derive the error sensitivity to spatial frequency function S_f. This function computes a normalised sensitivity based on a given spatial frequency f and a position $\mathbf{x}$ in the image. Figure A.1 shows the viewing geometry used for this derivation. For normal im age viewing, the observer's eye is at some distance V (measured in image widths) perpendicular to the image plane. In the case of VR goggles, this distance will be constant since the goggle display is always a fixed distance from the user's eyes. From Figure A.1, the eccentricity is given by the trigonometric relation in equation (A.2), where N is the image width:

$$e(d(\mathbf{x})) = tan^{-1}\left(\frac{d(\mathbf{x})}{NV}\right) \quad \text{(degrees)}. \tag{A.2}$$

Using equations (A.1) and (A.2), S_f is given by

$$\begin{aligned} S_f(f, d(\mathbf{x})) &= \frac{CS(f, e(d(\mathbf{x})))}{CS(f, 0)} \\ &= exp\Big(-0.0461 f \cdot e(d(\mathbf{x}))\Big). \end{aligned} \tag{A.3}$$

Due to the limits of the HVS and the VR goggle display resolution, there are restrictions to the acceptable values of S_f depending on the spatial frequency f. Using equation (A.1) and setting $CS(f, e) = 1$ (the maximum possible contrast sensitivity), the cutoff frequency f_c for a given e can be derived as equation (A.4). f_c is the highest spatial frequency

[1]The boldface denotes a 2D position vector.

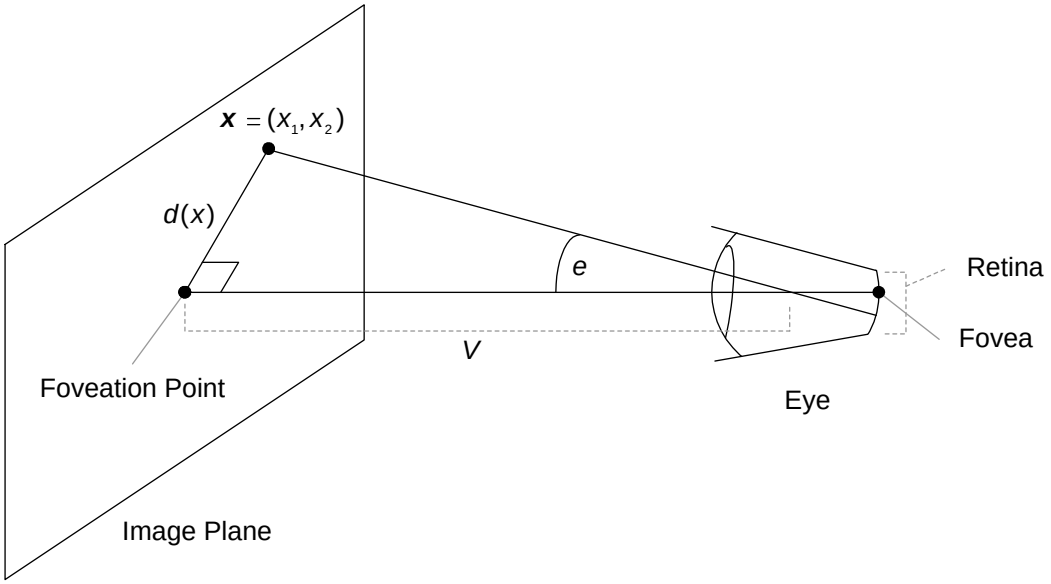


Figure A.1: Viewing geometry

which the human eye can detect. A similar limitation arises because the VR goggles can only output images at a certain resolution r (calculated by equation (A.5) and measured in pixels/degree). As shown in equation (A.6), the sampling theorem stipulates that the maximum spatial frequency f_d that can be displayed without aliasing effects is half the display resolution.

$$f_c = \frac{90.24}{e + 2.3} \quad \left(\frac{\text{cycles}}{\text{degree}}\right). \tag{A.4}$$

$$r = \frac{\pi NV}{180\cos^2(\frac{\pi e}{180})} = \frac{\pi NV}{180}\frac{N^2V^2}{d^2(\mathbf{x}) + N^2V^2} \approx \frac{\pi NV}{180} \quad \left(\frac{\text{pixels}}{\text{degree}}\right). \tag{A.5}$$

$$f_d = \frac{r}{2} \approx \frac{\pi NV}{360} \quad \left(\frac{\text{cycles}}{\text{degree}}\right). \tag{A.6}$$

Let $f_m(\mathbf{x}) = min(f_c, f_d)$, which gives the absolute highest spatial frequency at location $\mathbf{x}$ in the image that can be perceived by a human observer. Applying $f_m(\mathbf{x})$ to the sensitivity

equation (A.3) gives

$$S_f(f, d(\mathbf{x})) = \begin{cases} exp\Big(-0.0461 f \cdot e(d(\mathbf{x}))\Big) & \text{, for } f \leq f_m(\mathbf{x}) \\ 0 & \text{, for } f > f_m(\mathbf{x}) \end{cases} \tag{A.7}$$

The second step is to obtain a error sensitivity rating for each wavelet subband in the DWT. Again, psychological experiments have found a mathematical model that fits the observations of typical human viewers [40]. Given the subband level $\lambda \in 1, 2, 3...$ and orientation $\theta =$ LL, LH, HL, HH, the visual importance of each wavelet subband is computed using equation (A.8), where g_θ is 1.501, 1, 1 and 0.534 for the LL, LH, HL and HH subbands respectively. $A_{\lambda,\theta}$ is the wavelet basis function amplitude at a given λ and θ. For this experiment the 9/7 biorthogonal wavelet function is used for the DWT since it has been widely adopted for many wavelet-based compression algorithms [23, 27, 40]. Some basis function amplitudes for the 9/7 biorthogonal wavelet are given in Table A.1 and the corresponding $S_w(\lambda, \theta)$ values are given in Table A.2.

$$S_w(\lambda, \theta) = \frac{A_{\lambda,\theta}}{0.495 \cdot 10^{0.466(log_{10}(2^\lambda 0.401 g_\theta / r))^2}}, \tag{A.8}$$

Table A.1: Basis function amplitudes $A_{\lambda,\theta}$ for 9/7 biorthogonal DWT[1]

Orientation (θ)	Level (λ)					
	1	2	3	4	5	6
LL	0.62171	0.34537	0.18004	0.09140	0.045943	0.023013
LH	0.67234	0.41317	0.22727	0.11792	0.059758	0.030018
HL	0.67234	0.41317	0.22727	0.11792	0.059758	0.030018
HH	0.72709	0.49428	0.28688	0.15214	0.077727	0.039156

[1]Values reproduced from [59].

Before combining S_f and S_w to form the final wavelet domain sensitivity function, the

Table A.2: $S_w(\lambda, \theta)$ for V = 3, N = 256, 9/7 biorthogonal DWT

Orientation (θ)	Level (λ)					
	1	2	3	4	5	6
LL	0.3877	0.3842	0.2942	0.1806	0.0904	0.0371
LH	0.2728	0.3352	0.3035	0.2134	0.1207	0.0557
HL	0.2728	0.3352	0.3035	0.2134	0.1207	0.0557
HH	0.1333	0.2160	0.2461	0.2109	0.1433	0.0791

definition for $S_f(f, d(\mathbf{x}))$ must be slightly adjusted for the wavelet domain. Previously in equation (A.2), $d(\mathbf{x})$ was the distance, in the spatial domain, from the foveation point to the pixel at location $\mathbf{x}$. However, what is desired is for the final sensitivity function to be a function of $\mathbf{w}$, so a conversion needs to be performed to obtain the spatial $d(\mathbf{x})$ given a $\mathbf{w}$. Given $\mathbf{w}$, the distance from the foveation centre to the corresponding $\mathbf{x}$, in the spatial domain, depends on which subband $\mathbf{w}$ is in. Once the subband is known, the distance is computed by equation (A.9), where $\mathbf{w}^f_{\lambda,\theta}$ is the foveation centre location for the wavelet level λ and orientation θ.

$$d_{\lambda,\theta}(\mathbf{w}) = 2^{\lambda} \|\mathbf{w} - \mathbf{w}^f_{\lambda,\theta}\|, \tag{A.9}$$

where

$$\mathbf{w}^f_{\lambda,\theta} = \begin{cases} \left(\frac{x_1^f}{2^\lambda}, \frac{x_2^f}{2^\lambda}\right) & \text{, for } \theta = \text{LL} \\ \left(\frac{x_1^f + N}{2^\lambda}, \frac{x_2^f}{2^\lambda}\right) & \text{, for } \theta = \text{LH} \\ \left(\frac{x_1^f}{2^\lambda}, \frac{x_2^f + N}{2^\lambda}\right) & \text{, for } \theta = \text{HL} \\ \left(\frac{x_1^f + N}{2^\lambda}, \frac{x_2^f + N}{2^\lambda}\right) & \text{, for } \theta = \text{HH} \end{cases}$$

The final wavelet domain error sensitivity function $S(\mathbf{w})$ is derived from equations (A.7), (A.8) and (A.9) and is shown in equation (A.10), where β_1 and β_2 are used to control the magnitudes of S_w and S_f respectively. In this experiment $\beta_1 = 1$ and $\beta_2 = 2.5$, which are the same values used in [40]. Figure 3.5 in Section 3.3 shows a graphical representation of

$S(\mathbf{w})$ for V = 3 and N = 256. Dark areas are values near 0 and white areas are values at or near 1. A scaled version of this function is then applied as the foveated weighting mask.

$$S(\mathbf{w}) = [S_w(\lambda, \theta)]^{\beta_1} \cdot [S_f(r2^{-\lambda}, d_{\lambda,\theta}(\mathbf{w})]^{\beta_2} \tag{A.10}$$

Appendix B

Foveated Wavelet Quality Index Calculation

This appendix derives the foveated wavelet quality index (FWQI). The FWQI is given by equation (B.1).

$$FWQI = \frac{\sum_{\mathbf{w}} S(\mathbf{w})|c(\mathbf{w})|Q(\mathbf{w})}{\sum_{\mathbf{w}} S(\mathbf{w})|c(\mathbf{w})|}. \tag{B.1}$$

In equation (B.1), S($\mathbf{w}$) is the foveated sensitivity defined in Appendix A, $c(\mathbf{w})$ is the wavelet coefficient value at wavelet location $\mathbf{w}$ and $Q(\mathbf{w})$ is the quality index at wavelet location $\mathbf{w}$.[1] Q is calculated based on a combination of correlation, mean distortion and variance distortion [53]. Let $x = \{x_i | \, i = 1, 2, ..., N\}$ and $y = \{y_i | \, i = 1, 2, ..., N\}$ be the original and to-be-tested images respectively, and N be the constant number of pixels per image. Then Q is defined as

$$Q = \frac{4\sigma_{xy}\overline{x}\,\overline{y}}{(\sigma_x^2 + \sigma_y^2)[\overline{x}^2 + \overline{y}^2]}, \tag{B.2}$$

[1] The boldface denotes a 2D position vector.

where,

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \,,$$
$$\overline{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \,,$$
$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2 \,,$$
$$\sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \overline{y})^2 \,,$$
$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y}) \,.$$

Q is a 2D map that has values ranging between -1 and 1, with 1 representing a perfect match between the original image and the tested image. In turn, the FWQI ranges from 0 to 1, with 1 representing a perfect match.

Bibliography

[1] E. Dubois, "Video sampling and interpolation," in *Handbook of Image & Video Processing*, A. C. Bovik, Ed. London, United Kingdom: Academic Press, 2000, ch. 7.2.

[2] C.-W. Fung and S. C. Liew, "End-to-end frame-rate adaptive streaming of video data," in *Proc. IEEE International Conference on Multimedia Computing and Systems, 1999*, June 1999, pp. 67–71.

[3] J. Kim *et al.*, "TCP-friendly Internet video streaming employing variable frame-rate encoding and interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1164–1177, Oct. 2000.

[4] E. C. Reed and F. Dufaux, "Constrained bit-rate control for very low bit-rate streaming-video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 882–889, July 2001.

[5] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*. Berlin, Germany: Springer-Verlag, 2000.

[6] A. C. Bovik, "Introduction to digital image and video processing," in *Handbook of Image & Video Processing*, A. C. Bovik, Ed. London, United Kingdom: Academic Press, 2000, ch. 1.1.

[7] D. B. Diner and D. H. Fender, *Human Engineering in Stereoscopic Viewing Devices*. New York, New York, USA: Plenum Press, 1993.

[8] K. N. Ogle, *Researches in Binocular Vision*. Philadelphia, USA: W.B Saunders Company, 1950.

[9] P. J. Bos, "Performance limits of stereoscopic viewing systems using active and passive glasses," in *Proc. IEEE Virtual Reality Annual International Symposium, 1993*, Sept. 1993, pp. 371–376.

[10] K. M. Won S. Kim, Andrew Liu and L. Stark, "A helmet mounted display for telerobotics," in *Compcon Spring '88. Thirty-Third IEEE Computer Society International Conference, Digest of Papers*, Feb. 1988, pp. 543–547.

[11] C. F. Turner, "Development of an internet visual telepresence system," Master's thesis, University of Waterloo, Waterloo, 2002.

[12] S. P. Ken Perlin and J. S. Kollin, "An autostereoscopic display," in *Proc. ACM SIGGRAPH 2000 Conference Proceedings*, July 2000, pp. 319–326.

[13] J. S. Cagatay Basdogan, Mitchell Lum and E. Chow, "Autostereoscopic and haptic visualization for space exploration and mission design," in *Proc. IEEE 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002*, Mar. 2002, pp. 271–276.

[14] B. Barnett, "Basic concepts and techniques of video coding and the H.261 standard," in *Handbook of Image & Video Processing*, A. C. Bovik, Ed. London, United Kingdom: Academic Press, 2000, ch. 6.1.

[15] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Al-*

gorithms and Architectures–2nd ed. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1997.

[16] J. K. Shu, "Implementation of a force-reflecting teleoperation system over the internet," Master's thesis, University of Waterloo, Waterloo, 2001.

[17] C. Caradima, "Time delay compensation in teleoperation over the internet," Master's thesis, University of Waterloo, Waterloo, 1999.

[18] M. A. Sid-Ahmed, *Image Processing: Theory, Algorithms, & Architectures.* New York, New York, USA: McGraw-Hill, 1994.

[19] A. C. Bovik, Ed., *Handbook of Image & Video Processing.* London, United Kingdom: Academic Press, 2000.

[20] Z. Xiong and K. Ramchandran, "Wavelet image compression," in *Handbook of Image & Video Processing*, A. C. Bovik, Ed. London, United Kingdom: Academic Press, 2000, ch. 5.4.

[21] C. Blatter, *Wavelets: A Primer.* Natick, Massachusetts, USA: A K Peters, 1998.

[22] *Wavelets: A New Tool For Signal Analysis*, MathWorks Inc., in MATLAB 6.0 Help.

[23] P. M. Marc Antonini, Michel Barland and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.

[24] I. Daubechies, *Ten Lectures on Wavelets.* Philadelphia, Pennsylvania, USA: Society for Industrial and Applied Mathematics, 1992.

[25] T. Q. Nguyen, "A tutorial on filter banks and wavelets." [Online]. Available: citeseer.nj.nec.com/nguyen95tutorial.html

[26] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[27] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, Sept. 1996.

[28] Y.-P. H. Yong-Sheng Chen and C.-S. Fuh, "Fast block matching algorithm based on the winner-update strategy," *IEEE Trans. Image Processing*, vol. 10, pp. 1212–1222, Aug. 2001.

[29] Y.-Q. Z. Sohail Zafar and B. Jabbari, "Multiscale video representation using multiresolution motion compensation and wavelet decomposition," *IEEE J. Select. Areas Commun.*, vol. 11, pp. 24–35, Jan. 1993.

[30] S. Zafar and Y.-Q. Zhang, "Motion-compensated wavelet transform coding for color video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 285–296, Sept. 1992.

[31] C. Kim and J.-N. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 122–129, Feb. 2002.

[32] S. Cho and W. A. Pearlman, "A full-featured, error-resilient, scalable wavelet video codec based on the set partitioning in hierarchical trees (spiht) algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 157–171, Mar. 2002.

[33] S. S. M.W. Siegel, Priyan Gunatilake and A. Jordan, "Compression of stereo image pairs and streams," in *Proc. SPIE Stereoscopic Displays and Virtual Reality Systems*, Apr. 1994, pp. 258–268.

[34] M. G. Perkins, "Data compression of stereopairs," *IEEE Trans. Commun.*, vol. 40, pp. 684–696, Apr. 1992.

[35] J. R. I. Dinstein, G. Guy, "On stereo image coding," in *Proc. IEEE 9th International Conference on Pattern Recognition, 1988*, Nov. 1988, pp. 357–359.

[36] M. S. Moellenhoff and M. W. Maier, "Transform coding of stereo image residules," *IEEE Trans. Image Processing*, vol. 7, pp. 804–812, June 1998.

[37] Y. Zhang and G. Li, "An efficient hierarchical disparity estimation algorithm for stereoscopic video coding," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2000)*, Dec. 2000, pp. 744–747.

[38] J. Jiang and E. A. Edirisinghe, "A hybrid scheme for low bit-rate coding of stereo images," *IEEE Trans. Image Processing*, vol. 11, pp. 123–134, Feb. 2002.

[39] P. Kortum and W. Geisler, "Implementation of a foveated image coding system for image bandwidth reduction," in *Proc. SPIE Human Vision and Electronic Imaging*, Apr. 1996, pp. 350–360.

[40] Z. Wang and A. C. Bovik, "Embedded foveation image coding," *IEEE Trans. Image Processing*, vol. 10, pp. 1397–1410, Oct. 2001.

[41] T. H. Reeves, "Adaptive foveated video coding," Master's thesis, University of Waterloo, Waterloo, 1997.

[42] E.-C. Chang and C. K. Yap, "A wavelet approach to foveating images," in *Proc. 13th ACM Symposium on Computational Geometry*, Aug. 1997, pp. 397–399.

[43] I. I. Systems. (2002) Interactive imaging systems vfx3d product page. [Online]. Available: http://www.iisvr.com/products_VR_vfx3d.html

[44] A. Technologies. (2002) Ads technologies product page. [Online]. Available: http://www.adstech.com/products/PYRO1394WebCam/intro/PYRO1394WebCam.asp?pid=API-200

[45] Z. X. Beong-Jo Kim and W. A. Pearlman, "Low bit-rate scalable video coding with 3-d set partitioning in hierarchical trees (3-d spiht)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1374–1387, Dec. 2000.

[46] J. Gailly and M. Adler. (2002) zlib home site. [Online]. Available: http://www.gzip.org/zlib/

[47] C. F. Antonio Chimienti and D. Pau, "A complexity-bounded motion estimation algorithm," *IEEE Trans. Image Processing*, vol. 11, pp. 387–392, Apr. 2002.

[48] J. Chalidabhongse and C. J. Kuo, "Fast motion vector estimation using multiresolution spatio-temporal correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 477–488, June 1997.

[49] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Trans. Image Processing*, vol. 9, pp. 497–501, Mar. 2000.

[50] O. C. A. Wing Cheong Chan and M. F. Fu, "A novel predictive global motion estimation for video coding," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, May 2002, pp. 5–8.

[51] W. Zheng *et al.*, "A high-precision camera operation parameter measurement system and its application to image motion inferring," *IEEE Trans. Broadcast.*, vol. 47, pp. 46–55, Mar. 2001.

[52] A. C. B. Zhou Wang and L. Lu, "Why is image quality assessment so difficult?" in *Proc.*

IEEE International Conference on Acoustics, Speech, and Signal Processing, 2002, May 2002, pp. 3313–3316.

[53] L. L. Zhou Wang, Alan C. Bovik and J. Kouloheris, "Foveated wavelet image quality index," in *Proc. SPIE Applications of Digital Image Processing XXIV*, Dec. 2001, pp. 5–8.

[54] MpegTV. (2002) Mpeg.org - mpeg software simulation group(mss). [Online]. Available: http://www.mpeg.org/MPEG/MSSG/

[55] P. Corporation. (2002) Index of /ftp/video-site. [Online]. Available: http://standard.pictel.com/ftp/video-site

[56] G. Cote and L. Winger, "Recent advances in video compression standards," *IEEE Canadian Review - Spring/Printemps 2002*, pp. 21–24, 2002.

[57] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, pp. 186–200, 1996.

[58] W. S. Geisler and J. S. Perry, "Real-time foveated multiresolution system for low-bandwidth video communication," in *Proc. SPIE Human Vision and Electronic Imaging III*, July 1998, pp. 294–305.

[59] J. A. S. Andrea B. Watson, Gloria Y. Yang and J. Villasenor, "Visibility of wavelet quantization noise," *IEEE Trans. Image Processing*, vol. 6, pp. 1164–1175, Aug. 1997.